RM0506
Reference manual

STM32MP21xx advanced Arm®-based 32/64-bit MPUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32MP21xx microprocessor memory and peripherals.

The STM32MP21xx is a family of microprocessors with different packages and peripherals.

Refer to the corresponding datasheets for ordering information, mechanical and electrical device characteristics.

For information on the Arm® Cortex®-A35 and Cortex®-M33 cores, refer to the corresponding Technical Reference Manuals available on http://infocenter.arm.com.

The STM32MP21xx microprocessors include ST state-of-the-art patented technology.

Related documents

- Cortex®-A35 Technical Reference Manual, available from: http://infocenter.arm.com
- Cortex®-M33 Technical Reference Manual, available from: http://infocenter.arm.com
- STM32MP21xA/D datasheet (DS14557)
- STM32MP21xC/F datasheet (DS14556)
- STM32 Cortex®-M33 MCUs programming manual (PM0264)
- STM32MP21xA/C/D/F device errata (ES0628)

Contents

1 Documentation conventions 167

1.1 General information 167

1.2 List of abbreviations for registers 167

1.3 Register reset value 168

1.4 Glossary 168

1.5 Availability of peripherals 168

2 Memory and bus architecture 169

2.1 System architecture 169

2.1.1 Bus architecture 169

2.1.2 Memory map organization 172

2.1.3 Connectivity matrix overview 174

2.2 STNoC AXI network-on-chip 175

2.2.1 STNoC features 175

2.2.2 STNoC interconnect configurations 175

2.2.3 Master port description 177

2.2.4 Master port main characteristics 178

2.2.5 Slave port description 179

2.2.6 Slave port main characteristics 179

2.3 Multi-layer AHB interconnect 180

2.3.1 Multi-layer AHB features 180

2.3.2 Multi-layer AHB interconnect configuration 180

2.3.3 Multi-layer AHB 300 MHz master port characteristics 181

2.3.4 Multi-layer AHB 300 MHz slave port characteristics 181

2.3.5 Multi-layer AHB 150 MHz master port characteristics 182

2.3.6 Multi-layer AHB 150 MHz slave port characteristics 182

2.4 Memory organization 184

2.4.1 Introduction 184

2.4.2 Memory map and register boundary addresses 185

3 Arm Cortex-A35 (CA35SS) 203

3.1 Overview 203

3.2 Cortex-A35 reset 204

3.3 Cortex-A35 clocking 204
3.4 CortexA35 boot mode 204
3.5 CA35SS SYSCFG registers 205
3.5.1 System debug configuration registers 205
3.5.2 Cortex-A35 subsystem configuration registers 209
3.5.3 Cortex-A35 boot configuration registers 211
3.5.4 Cortex-M33 boot configuration registers 213
3.5.5 CA35SS SYSCFG register map 215
3.6 Standardized status and control registers 217
3.6.1 Standardized status and control registers (SSC) access modes 217
3.6.2 SSC access types 217
3.6.3 Secure/non-secure access to SSC 218
3.6.4 SSC registers details 218
3.6.5 CA35SS SSC register map 225

4 Arm Cortex-M33 (CM33SS) 228
4.1 Overview 228
4.2 Cortex-M33 reset 230
4.3 Cortex-M33 clocking 230
4.4 Cortex-M33 boot mode 230
4.5 Cortex-M33 configuration register 230
4.6 Memory protection unit (MPU) 231
4.7 Floating point unit (FPU) 231
4.8 Attribution units 231
4.8.1 Security attribution unit (SAU) 231
4.8.2 Implementation defined attribution unit (IDAU) 231
4.9 Nested vectored interrupt controllers (NVIC) 232
4.9.1 NVIC features 232
4.9.2 Interrupt and exception vectors 232
4.10 Cortex-M33 memory interfaces 232

5 System security 234
5.1 Key security features 234
5.2 Secure boot 235
5.3 Secure provisioning 236

5.4 Resource isolation framework (RIF) 236
5.4.1 RIF security architecture 237
5.4.2 RIF access control 239
5.4.3 Application interrupt management in RIF 241
5.4.4 RIF illegal access definition 242
5.4.5 Managing illegal accesses with RIF 244
5.4.6 RIF special security domain: trusted domain 244
5.4.7 RIF special security domain: debug domain 245
5.4.8 Managing security in RIF-aware peripherals 245
5.4.9 Activating RIF security 246
5.4.10 Programming of RIFSC, RCC and IAC 247

6 Boot and security control (BSEC) 248
6.1 Introduction 248
6.2 BSEC main features 249
6.3 BSEC functional description 250
6.3.1 BSEC block diagram 250
6.3.2 BSEC internal signals 250
6.3.3 BSEC reset and clocks 251
6.3.4 Organization and use of fuses 251
6.3.5 Operations on fuses 253
6.3.6 BSEC read and programming status reporting 255
6.3.7 Lifecycle management 256
6.3.8 Epoch registers 258
6.3.9 BSEC local access control 258
6.3.10 Temporal isolation in BSEC 258
6.3.11 BSEC debug and trace control 260
6.3.12 Device features control 262
6.3.13 Scratch registers 262
6.3.14 JTAG registers 262
6.3.15 STM32 root hardware unique key (RHUK) 262
6.3.16 BSEC error management 263
6.3.17 BSEC tamper response mechanisms 263
6.4 BSEC interrupts 263
6.5 BSEC registers 263
6.5.1 BSEC fuse word w value register (BSEC_FVRw) 264

6.5.2 BSEC sticky programming lock register x (BSEC_SPLOCKx) 265
6.5.3 BSEC sticky write lock register x (BSEC_SWLOCKx) 265
6.5.4 BSEC sticky reload lock register x (BSEC_SRLOCKx) 266
6.5.5 BSEC OTP valid register x (BSEC_OTPVLDRx) 266
6.5.6 BSEC shadowed fuses status register x (BSEC_SFSRx) 267
6.5.7 BSEC OTP control register (BSEC_OTPCR) 267
6.5.8 BSEC write data register (BSEC_WDR) 268
6.5.9 BSEC scratch register x (BSEC_SCRATCHRx) 269
6.5.10 BSEC lock register (BSEC_LOCKR) 269
6.5.11 BSEC JTAG input register (BSEC_JTAGINR) 270
6.5.12 BSEC JTAG output register (BSEC_JTAGOUTR) 270
6.5.13 BSEC debug enable register (BSEC_DENR) 271
6.5.14 BSEC unmap register (BSEC_UNMAPR) 272
6.5.15 BSEC status register (BSEC_SR) 273
6.5.16 BSEC OTP status register (BSEC_OTPSR) 274
6.5.17 BSEC epoch register (BSEC_EPOCHRx) 275
6.5.18 BSEC epoch selection control register (BSEC_EPOCHSELCR) 276
6.5.19 BSEC debug control register for Cortex-M (BSEC_DBGMCR) 276
6.5.20 BSEC AP unlock (BSEC_AP_UNLOCK) 277
6.5.21 BSEC HDPL for Cortex-M status register (BSEC_HDPLMSR) 278
6.5.22 BSEC HDPL for Cortex-M control register (BSEC_HDPLMCR) 278
6.5.23 BSEC next HDPL control register for Cortex-M (BSEC_NEXTHDPLMCR) 279
6.5.24 BSEC debug control register for Cortex-A (BSEC_DBGACR) 279
6.5.25 BSEC HDPL for Cortex-A status register (BSEC_HDPLASR) 280
6.5.26 BSEC HDPL for Cortex-A control register (BSEC_HDPLACR) 281
6.5.27 BSEC next HDPL control register for Cortex-A (BSEC_NEXTHDPLACR) 281
6.5.28 BSEC write once scratch register x (BSEC_WOSCRx) 282
6.5.29 BSEC hot reset count register (BSEC_HRCR) 282
6.5.30 BSEC warm reset count register (BSEC_WRCR) 283
6.5.31 BSEC version register (BSEC_VERR) 283
6.5.32 BSEC identification register (BSEC_IPIDR) 284
6.5.33 BSEC size identification register (BSEC_SIDR) 284
6.5.34 BSEC register map 284

7 OTP mapping (OTP) 288

8 Resource isolation framework security controller (RIFSC) 307

8.1 Introduction 307

8.2 RIFSC main features 307

8.3 RIFSC implementation 308

8.4 RIFSC functional description 310

8.4.1 RIFSC block diagrams 310

8.4.2 RIFSC internal signals 312

8.4.3 RIFSC reset and clocks 312

8.4.4 RIFSC control of masters (RIMU) 313

8.4.5 RIFSC control of debug access port (DAP) 313

8.4.6 RIFSC control of peripherals (RISUP) 315

8.4.7 Managing illegal accesses 317

8.5 RIFSC registers 317

8.5.1 RIFSC RISC slave configuration register x (RIFSC_RISC_CR) 318

8.5.2 RIFSC RISC slave security configuration register x (RIFSC_RISC_SECCFGRx) 318

8.5.3 RIFSC RISFC slave privileged register x (RIFSC_RISC_PRIVCFGRx) 319

8.5.4 RIFSC RISC slave resource configuration lock register x (RIFSC_RISC_RCFGLOCKRx) 319

8.5.5 RIFSC RISC slave peripheral y CID configuration register (RIFSC_RISC_PERy_CIDCFGR) 320

8.5.6 RIFSC RISC slave peripheral y semaphore control register (RIFSC_RISC_PERy_SEMCR) 321

8.5.7 RIFSC RIMC master configuration register (RIFSC_RIMC_CR) 322

8.5.8 RIFSC RIMC master status register (RIFSC_RIMC_SR) 323

8.5.9 RIFSC RIMC master attribute register x (RIFSC_RIMC_ATTRx) 324

8.5.10 RIFSC peripheral protection status register x (RIFSC_PPSRx) 325

8.5.11 RIFSC hardware configuration register 3 (RIFSC_HWCFGR3) 325

8.5.12 RIFSC hardware configuration register 2 (RIFSC_HWCFGR2) 325

8.5.13 RIFSC hardware configuration register 1 (RIFSC_HWCFGR1) 326

8.5.14 RIFSC version register (RIFSC_VERR) 326

8.5.15 RIFSC identification register (RIFSC_IPIDR) 327

8.5.16 RIFSC size identification register (RIFSC_SIDR) 327

8.5.17 RIFSC register map 327

9 Resource isolation slave unit for address space protection (block-based) (RISAB) 330

9.1 RISAB introduction 330
9.2 RISAB main features 330
9.3 RISAB implementation 330
9.4 RISAB functional description 331
9.4.1 RISAB block diagram 331
9.4.2 RISAB internal signals 331
9.4.3 RISAB reset and clocks 331
9.4.4 RISAB address space management 332
9.4.5 RISAB programming 332
9.4.6 RISAB runtime modification of region or subregion configuration 333
9.4.7 Managing illegal accesses 333
9.5 RISAB registers 334
9.5.1 RISAB configuration register (RISAB_CR) 334
9.5.2 RISAB illegal access status register (RISAB_IASR) 334
9.5.3 RISAB illegal access clear register (RISAB_IACR) 335
9.5.4 RISAB configuration lock register (RISAB_RCFGLOCKR) 335
9.5.5 RISAB illegal access error status register (RISAB_IAESR) 336
9.5.6 RISAB illegal address register (RISAB_IADDR) 337
9.5.7 RISAB page y security configuration register (RISAB_PGy_SECCFGR) 337
9.5.8 RISAB page y default privileged configuration register (RISAB_PGy_PRIVCFGR) 338
9.5.9 RISAB page y privileged configuration register for compartment 2 (RISAB_PGy_C2PRIVCFGR) 338
9.5.10 RISAB compartment x privilege configuration register (RISAB_CIDxPRIVCFGR) 339
9.5.11 RISAB compartment x read configuration register (RISAB_CIDxRDCFGR) 340
9.5.12 RISAB compartment x write configuration register (RISAB_CIDxWRCFGR) 340
9.5.13 RISAB page y CID configuration register (RISAB_PGy_CIDCFGR) . . 341
9.5.14 RISAB hardware configuration register 3 (RISAB_HWCFGR3) 342
9.5.15 RISAB hardware configuration register 2 (RISAB_HWCFGR2) 342
9.5.16 RISAB hardware configuration register 1 (RISAB_HWCFGR1) 343
9.5.17 RISAB version register (RISAB_VERR) 343
9.5.18 RISAB identification register (RISAB_IPIDR) 344
9.5.19 RISAB size identification register (RISAB_SIDR) 344
9.5.20 RISAB register map 345

10 Resource isolation slave unit for address space protection (full version) (RISAF) 347

10.1 RISAF introduction 347

10.2 RISAF main features 347

10.3 RISAF implementation 348

10.4 RISAF functional description 348

10.4.1 RISAF block diagram 348

10.4.2 RISAF internal signals 349

10.4.3 RISAF reset and clocks 349

10.4.4 RISAF address space management 349

10.4.5 RISAF programming 351

10.4.6 RISAF runtime modification of region or subregion configuration 353

10.4.7 Managing illegal accesses 354

10.5 RISAF registers 355

10.5.1 RISAF configuration register (RISAF_CR) 355

10.5.2 RISAF status register (RISAF_SR) 355

10.5.3 RISAF illegal access status register (RISAF_IASR) 356

10.5.4 RISAF illegal access clear register (RISAF_IACR) 357

10.5.5 RISAF illegal access error status register x (RISAF_IAESRx) 357

10.5.6 RISAF illegal address register x (RISAF_IADDRx) 358

10.5.7 RISAF encryption key y (RISAF_KEYRy) 358

10.5.8 RISAF region x configuration register (RISAF_REGx_CFGR) 359

10.5.9 RISAF region x start-address register (RISAF_REGx_STARTR) 360

10.5.10 RISAF region x end-address register (RISAF_REGx_ENDR) 361

10.5.11 RISAF region x CID configuration register (RISAF_REGx_CIDCFGR) 361

10.5.12 RISAF region x subregion A configuration register (RISAF_REGx_ACFGR) 362

10.5.13 RISAF region x subregion A start-address register (RISAF_REGx_ASTARTR) 364

10.5.14 RISAF region x subregion A end-address register (RISAF_REGx_AENDR) 364

10.5.15 RISAF region x subregion A nested mode register (RISAF_REGx_ANESTR) 365

10.5.16 RISAF region x subregion B configuration register (RISAF_REGx_BCFGR) 366

10.5.17 RISAF region x subregion B start-address register (RISAF_REGx_BSTARTR) 367

10.5.18 RISAF region x subregion B end-address register (RISAF_REGx_BENDR) . . . 368
10.5.19 RISAF region x subregion B nested mode register (RISAF_REGx_BNESTR) . . . 369
10.5.20 RISAF hardware configuration register (RISAF_HWCFGR) . . . 369
10.5.21 RISAF version register (RISAF_VERR) . . . 370
10.5.22 RISAF identification register (RISAF_IPIDR) . . . 370
10.5.23 RISAF size identification register (RISAF_SIDR) . . . 371
10.5.24 RISAF extended configuration register (RISAF_XCR) . . . 371
10.5.25 RISAF extended status register (RISAF_XSR) . . . 372
10.5.26 RISAF master key y (RISAF_MKEYRy) . . . 373
10.5.27 RISAF register map . . . 373

11 Illegal access controller (IAC) . . . 376

11.1 IAC introduction . . . 376
11.2 IAC main features . . . 376
11.3 IAC implementation . . . 376
11.4 IAC functional description . . . 376
11.4.1 IAC block diagram . . . 376
11.4.2 IAC internal signals . . . 377
11.4.3 IAC reset and clocks . . . 377
11.4.4 IAC use in RIF . . . 377
11.4.5 IAC management by trusted application . . . 377
11.5 IAC interrupts . . . 378
11.6 IAC registers . . . 378
11.6.1 IAC interrupt enable register x (IAC_IERx) . . . 378
11.6.2 IAC interrupt status register x (IAC_ISRx) . . . 379
11.6.3 IAC interrupt clear register x (IAC_ICRx) . . . 379
11.6.4 IAC ILAC input status register x (IAC_IISRx) . . . 380
11.6.5 IAC hardware configuration register 2 (IAC_HWCFGR2) . . . 380
11.6.6 IAC hardware configuration register 1 (IAC_HWCFGR1) . . . 380
11.6.7 IAC version register (IAC_VERR) . . . 381
11.6.8 IAC identification register (IAC_IPIDR) . . . 381
11.6.9 IAC size identification register (IAC_SIDR) . . . 382
11.6.10 IAC register map . . . 382

12 DDR subsystem architecture (DDRSS) . . . 383

12.1 DDRS introduction . . . 383
12.2 DDRSS block diagram . . . 384
12.3 DDRSS interfaces . . . 384
12.4 DDRSS embedded fractional PLL2 . . . 385
12.5 DDRSS top-level features . . . 386
12.5.1 Filtering and AES encryption . . . 386
12.5.2 DDR3/4 AC-swizzling . . . 387
12.5.3 Mode register read . . . 388
12.5.4 Observability . . . 389
12.5.5 DDRSS low power . . . 391
12.5.6 DFI errors . . . 392
12.6 DDRSS registers . . . 393
12.6.1 DDRDBG low-power disable register (DDRDBG_LP_DISABLE) . . . 393
12.6.2 DDRDBG bypass P_{CLKEN} register (DDRDBG_BYPASS_PCLKEN) . . 393
12.6.3 DDRDBG DDRC auto SR delay register (DDRDBG_DDRC_AUTO_SR_DELAY) . . . 394
12.6.4 DDRDBG OBSP control register (DDRDBG_OBSP_CTRL) . . . 394
12.6.5 DDRDBG FRAC PLL lock register (DDRDBG_FRAC_PLL_LOCK) . . . 395
12.6.6 DDRDBG DTO output register (DDRDBG_DTO_OUTPUT) . . . 395
12.6.7 DDRDBG DFI error control register (DDRDBG_DFI_ERRORS_CTRL) . . . 396
12.6.8 DDRDBG DFI error status register (DDRDBG_DFI_ERROR_STATUS) . . . 396
12.6.9 DDRDBG DFI error mask register (DDRDBG_DFI_ERROR_MSK) . . . 397
12.6.10 DDRDBG DDR34 AC swizzle ADD3_0 register (DDRDBG_DDR34_AC_SWIZZLE_ADD3_0) . . . 397
12.6.11 DDRDBG DDR34 AC swizzle ADD7_4 register (DDRDBG_DDR34_AC_SWIZZLE_ADD7_4) . . . 399
12.6.12 DDRDBG DDR34 AC swizzle ADD11_8 register (DDRDBG_DDR34_AC_SWIZZLE_ADD11_8) . . . 399
12.6.13 DDRDBG DDR34 AC swizzle ADD15_12 register (DDRDBG_DDR34_AC_SWIZZLE_ADD15_12) . . . 400
12.6.14 DDRDBG DDR34 AC swizzle BK2 0 ACTN register (DDRDBG_DDR34_AC_SWIZZLE_BK2_0_ACTN) . . . 400
12.6.15 DDRDBG DDR34 AC swizzle RASN CASN BG1_0 register (DDRDBG_DDR34_AC_SWIZZLE_RASN_CASN_BG1_0) . . . 401
12.6.16 DDRDBG DDR34 AC swizzle WEN register (DDRDBG_DDR34_AC_SWIZZLE_WEN) . . . 402

12.6.17 DDRSS register map 402

13 DDR3L/DDR4/LPDDR4 controller (DDRCTRL) 405

13.1 DDRCTRL introduction 405
13.2 DDRCTRL features 405
13.3 DDRCTRL block diagram 406
13.4 DDRCTRL architecture overview 406
13.4.1 AXI port interface (XPI) 407
13.4.2 Port arbiter (PA) 408
13.4.3 Host interface (HIF) 408
13.4.4 DDR scheduler (DDRC) 409
13.5 DDRCTRL functional description 409
13.5.1 Transaction service control (TSC) and quality of service (QoS) 409
13.5.2 Paging policy options 410
13.5.3 Power saving features 411
13.5.4 Address mapper 411
13.5.5 DRAM timing parameters 412
13.5.6 SDRAM initialization sequence 412
13.5.7 Refresh controls 412
13.5.8 ZQ calibration 413
13.5.9 Register programming 413
13.6 DDRCTRL configuration 414
13.7 DDRCTRL registers 415
13.7.1 DDRCTRL master0 register (DDRCTRL_REGS_MSTR) 415
13.7.2 DDRCTRL operating mode status register (DDRCTRL_REGS_STAT) 417
13.7.3 DDRCTRL mode read/write control 0 register (DDRCTRL_REGS_MRCTRL0) 418
13.7.4 DDRCTRL mode read/write control 1 register (DDRCTRL_REGS_MRCTRL1) 420
13.7.5 DDRCTRL mode register read/write status register (DDRCTRL_REGS_MRSTAT) 421
13.7.6 DDRCTRL mode register read/write control 2 register (DDRCTRL_REGS_MRCTRL2) 421
13.7.7 DDRCTRL temperature derate enable register (DDRCTRL_REGS_DERATEEN) 422
13.7.8 DDRCTRL temperature derate interval register (DDRCTRL_REGS_DERATEINT) 423

13.7.9 DDRCTRL temperature derate control register (DDRCTRL_REGS_DERATECTL) 424
13.7.10 DDRCTRL low-power control register (DDRCTRL_REGS_PWRCTL) 424
13.7.11 DDRCTRL low-power timing register (DDRCTRL_REGS_PWRTMG) 426
13.7.12 DDRCTRL hardware low-power control register (DDRCTRL_REGS_HWLPCTL) 426
13.7.13 DDRCTRL refresh control 0 register (DDRCTRL_REGS_RFSHCTL0) 427
13.7.14 DDRCTRL refresh control 1 register (DDRCTRL_REGS_RFSHCTL1) 429
13.7.15 DDRCTRL refresh control 3 register (DDRCTRL_REGS_RFSHCTL3) 429
13.7.16 DDRCTRL refresh timing register (DDRCTRL_REGS_RFSHTMG) . . 431
13.7.17 DDRCTRL refresh timing1 register (DDRCTRL_REGS_RFSHTMG1) 432
13.7.18 DDRCTRL SDRAM initialization 0 register (DDRCTRL_REGS_INIT0) 433
13.7.19 DDRCTRL SDRAM initialization 1 register (DDRCTRL_REGS_INIT1) 434
13.7.20 DDRCTRL SDRAM initialization 3 register (DDRCTRL_REGS_INIT3) 434
13.7.21 DDRCTRL SDRAM initialization 4 register (DDRCTRL_REGS_INIT4) 435
13.7.22 DDRCTRL SDRAM initialization 5 register (DDRCTRL_REGS_INIT5) 435
13.7.23 DDRCTRL SDRAM initialization 6 register (DDRCTRL_REGS_INIT6) 436
13.7.24 DDRCTRL SDRAM initialization 7 register (DDRCTRL_REGS_INIT7) 436
13.7.25 DDRCTRL rank control register (DDRCTRL_REGS_RANKCTL) 437
13.7.26 Rank control 1 register (DDRCTRL_REGS_RANKCTL1) 439
13.7.28 DDRCTRL SDRAM timing 1 register (DDRCTRL_REGS_DRAMTMG1) 441
13.7.29 DDRCTRL SDRAM timing 2 register (DDRCTRL_REGS_DRAMTMG2) 442
13.7.30 DDRCTRL SDRAM timing 3 register (DDRCTRL_REGS_DRAMTMG3) 444
13.7.31 DDRCTRL SDRAM timing 4 register (DDRCTRL_REGS_DRAMTMG4) 445
13.7.32 DDRCTRL SDRAM timing 5 register (DDRCTRL_REGS_DRAMTMG5) 446
13.7.33 DDRCTRL SDRAM timing 6 register (DDRCTRL_REGS_DRAMTMG6) 447
13.7.34 DDRCTRL SDRAM timing 7 register (DDRCTRL_REGS_DRAMTMG7) 448
13.7.35 DDRCTRL SDRAM timing 8 register (DDRCTRL_REGS_DRAMTMG8) 449

13.7.36 DDRCTRL SDRAM timing 9 register (DDRCTRL_REGS_DRAMTMG9) 450
13.7.37 DDRCTRL SDRAM timing 10 register (DDRCTRL_REGS_DRAMTMG10) 451
13.7.38 DDRCTRL SDRAM timing 11 register (DDRCTRL_REGS_DRAMTMG11) 452
13.7.39 DDRCTRL SDRAM timing 12 register (DDRCTRL_REGS_DRAMTMG12) 453
13.7.40 DDRCTRL SDRAM timing 13 register (DDRCTRL_REGS_DRAMTMG13) 454
13.7.41 DDRCTRL SDRAM timing 14 register (DDRCTRL_REGS_DRAMTMG14) 454
13.7.42 DDRCTRL SDRAM timing 15 register (DDRCTRL_REGS_DRAMTMG15) 455
13.7.43 DDRCTRL ZQ control 0 register (DDRCTRL_REGS_ZQCTL0) 456
13.7.44 DDRCTRL ZQ control 1 register (DDRCTRL_REGS_ZQCTL1) 457
13.7.45 DDRCTRL ZQ control 2 register (DDRCTRL_REGS_ZQCTL2) 457
13.7.46 DDRCTRL ZQ status register (DDRCTRL_REGS_ZQSTAT) 458
13.7.47 DDRCTRL DFI timing 0 register (DDRCTRL_REGS_DFITMG0) 459
13.7.48 DDRCTRL DFI timing 1 register (DDRCTRL_REGS_DFITMG1) 460
13.7.49 DDRCTRL DFI low-power configuration 0 register (DDRCTRL_REGS_DFILPCFG0) 461
13.7.50 DDRCTRL DFI update 0 register (DDRCTRL_REGS_DFIUPD0) 463
13.7.51 DDRCTRL DFI update 1 register (DDRCTRL_REGS_DFIUPD1) 464
13.7.52 DDRCTRL DFI update 2 register (DDRCTRL_REGS_DFIUPD2) 465
13.7.53 DDRCTRL DFI miscellaneous control register (DDRCTRL_REGS_DFIMISC) 465
13.7.54 DDRCTRL DFI timing 2 register (DDRCTRL_REGS_DFITMG2) 466
13.7.55 DDRCTRL DFI timing 3 register (DDRCTRL_REGS_DFITMG3) 467
13.7.56 DDRCTRL DFI status register (DDRCTRL_REGS_DFISTAT) 467
13.7.57 DDRCTRL DM/DBI control register (DDRCTRL_REGS_DBICTL) . . . 468
13.7.58 DDRCTRL DFI PHY master register (DDRCTRL_REGS_DFIPHYMSTR) 469
13.7.59 DDRCTRL address map 0 register (DDRCTRL_REGS_ADDRMAP0) 469
13.7.60 DDRCTRL address map 1 register (DDRCTRL_REGS_ADDRMAP1) 470
13.7.61 DDRCTRL address map 2 register (DDRCTRL_REGS_ADDRMAP2) 471
13.7.62 DDRCTRL address map 3 register (DDRCTRL_REGS_ADDRMAP3) 472
13.7.63 DDRCTRL address map 4 register (DDRCTRL_REGS_ADDRMAP4) 474
13.7.64 DDRCTRL address map 5 register (DDRCTRL_REGS_ADDRMAP5) 475
13.7.65 DDRCTRL address map 6 register (DDRCTRL_REGS_ADDRMAP6) 476

13.7.66 DDRCTRL address map 7 register (DDRCTRL_REGS_ADDRMAP7) 477
13.7.67 DDRCTRL address map 8 register (DDRCTRL_REGS_ADDRMAP8) 478
13.7.68 DDRCTRL address map 9 register (DDRCTRL_REGS_ADDRMAP9) 478
13.7.69 DDRCTRL address map 10 register (DDRCTRL_REGS_ADDRMAP10) 480
13.7.70 DDRCTRL address map 11 register (DDRCTRL_REGS_ADDRMAP11) 481
13.7.71 DDRCTRL ODT configuration register (DDRCTRL_REGS_ODTCFG) 481
13.7.72 DDRCTRL ODT/rank map register (DDRCTRL_REGS_ODTMAP) . . 483
13.7.73 DDRCTRL scheduler control register (DDRCTRL_REGS_SCHED) . . 484
13.7.74 DDRCTRL scheduler control 1 register (DDRCTRL_REGS_SCHED1) 486
13.7.75 DDRCTRL high priority read CAM 1 register (DDRCTRL_REGS_PERFHPR1) 488
13.7.76 DDRCTRL low-priority read CAM 1 register (DDRCTRL_REGS_PERFLPR1) 489
13.7.77 DDRCTRL write CAM 1 register (DDRCTRL_REGS_PERFWR1) . . . 489
13.7.78 Scheduler control 3 register (DDRCTRL_REGS_SCHED3) 490
13.7.79 Scheduler control 4 register (DDRCTRL_REGS_SCHED4) 491
13.7.80 DDRCTRL debug 0 register (DDRCTRL_REGS_DBG0) 492
13.7.81 DDRCTRL debug 1 register (DDRCTRL_REGS_DBG1) 493
13.7.82 DDRCTRL CAM debug register (DDRCTRL_REGS_DBGCAM) 493
13.7.83 DDRCTRL command debug register (DDRCTRL_REGS_DBGCMD) 495
13.7.84 DDRCTRL status debug register (DDRCTRL_REGS_DBGSTAT) . . . 496
13.7.85 DDRCTRL software programming control enable register (DDRCTRL_REGS_SWCTL) 497
13.7.86 DDRCTRL software programming control status register (DDRCTRL_REGS_SWSTAT) 497
13.7.87 DDRCTRL statics write enable register (DDRCTRL_REGS_SWCTLSTATIC) 498
13.7.88 DDRCTRL AXI poison configuration register (DDRCTRL_REGS_POISONCFG) 498
13.7.89 DDRCTRL AXI poison status register (DDRCTRL_REGS_POISONSTAT) 499
13.7.90 DDRCTRL temperature derate status register (DDRCTRL_REGS_DERATESTAT) 500
13.7.91 DDRCTRL port status register (DDRCTRL_MP_PSTAT) 501
13.7.92 DDRCTRL port common configuration register (DDRCTRL_MP_PCCFG) 501
13.7.93 DDRCTRL port x configuration read register (DDRCTRL_MP_PCFGR_x) 502

13.7.94 DDRCTRL port x configuration write register (DDRCTRL_MP_PCFGW_x) . . . 504
13.7.95 DDRCTRL port x control register (DDRCTRL_MP_PCTRL_x) . . . 505
13.7.96 DDRCTRL port x read QoS configuration 0 register (DDRCTRL_MP_PCFGQOS0_x) . . . 505
13.7.97 DDRCTRL port x read QoS configuration 1 register (DDRCTRL_MP_PCFGQOS1_x) . . . 506
13.7.98 DDRCTRL port x write QoS configuration 0 register (DDRCTRL_MP_PCFGWQOS0_x) . . . 506
13.7.99 DDRCTRL port x write QoS configuration 1 register (DDRCTRL_MP_PCFGWQOS1_x) . . . 508
13.7.100 DDRCTRL version number register (DDRCTRL_MP_DDRCTRL_VER_NUMBER) . . . 508
13.7.101 DDRCTRL version type register (DDRCTRL_MP_DDRCTRL_VER_TYPE) . . . 509
13.7.102 DDRCTRL register map . . . 509

14 DDR physical interface control (DDRPHYC) . . . 526

14.1 DDRPHYC features . . . 526
14.2 DDRPHYC block diagram . . . 526
14.3 DDRPHYC SDRAM interface . . . 527
14.4 DDRPHY functional overview . . . 529
14.4.1 MASTER . . . 529
14.4.2 ACX4 . . . 530
14.4.3 DBYTE . . . 530
14.4.4 PHY clocking . . . 531
14.4.5 PUB (PHY utility block) . . . 531
14.4.6 PHY and SDRAM initialization . . . 532
14.4.7 DRAM initialization sequence . . . 533
14.4.8 Device initialization . . . 534
14.4.9 Command address (CA) training . . . 534
14.4.10 Read gate training . . . 534
14.4.11 Write leveling fine phase training . . . 534
14.4.12 Read deskew . . . 535
14.4.13 Read 1D (SI friendly) training . . . 535
14.4.14 Write leveling coarse delay training . . . 535
14.4.15 Write 1D training . . . 536
14.4.16 Read 1D training (optimized delay centering) . . . 536
14.4.17 Max read latency training . . . 536

14.5 DDRPHYC registers 537
14.5.1 DDRPHYC MtestMuxSel register (DDRPHYC_ANIBx_MTESTMUXSEL) 537
14.5.2 DDRPHYC AForceDrvCont register (DDRPHYC_ANIBx_AFORCEDRVCONT) 538
14.5.3 DDRPHYC AForceTriCont register (DDRPHYC_ANIBx_AFORCETRICONT) 538
14.5.4 DDRPHYC address Tx impedance control register (DDRPHYC_ANIBx_ATXIMPEDANCE) 539
14.5.5 DDRPHYC ATestPrbsErr register (DDRPHYC_ANIBx_ATESTPRBSERR) 539
14.5.6 DDRPHYC address Tx slew rate and predriver control register (DDRPHYC_ANIBx_ATXSLEWRATE) 540
14.5.7 DDRPHYC ATestPrbsErrCnt register (DDRPHYC_ANIBx_ATESTPRBSERRCNT) 541
14.5.8 DDRPHYC ATxDly register (DDRPHYC_ANIBx_ATXDLY) 541
14.5.9 DDRPHYC DBYTE module disable register (DDRPHYC_DBYTEx_DBYTEMISCMODE) 542
14.5.10 DDRPHYC MtestMuxSel register (DDRPHYC_DBYTEx_MTESTMUXSEL) 542
14.5.11 DDRPHYC DFIMRL register (DDRPHYC_DBYTEx_DFIMRL) 543
14.5.12 DDRPHYC VrefDAC0_ry register (DDRPHYC_DBYTEx_VREFDAC0_Ry) 543
14.5.13 DDRPHYC data Tx impedance controls register (DDRPHYC_DBYTEx_TXIMPEDANCECTRL0_By) 544
14.5.14 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL_By) 545
14.5.15 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL1) 546
14.5.16 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL2) 546
14.5.17 DDRPHYC TX ODT driver strength control register (DDRPHYC_DBYTEx_TXODTDRVSTREN_By) 547
14.5.18 DDRPHYC status of RX FIFO consistency checks register (DDRPHYC_DBYTEx_RXFIFOCHECKSTATUS) 548
14.5.19 DDRPHYC captured values associated with an RxFIFO consistency error register (DDRPHYC_DBYTEx_RXFIFOCHECKERRVALUES) . 549
14.5.20 DDRPHYC data receive FIFO pointer values register (DDRPHYC_DBYTEx_RXFIFOINFO) 549
14.5.21 DDRPHYC Rx FIFO visibility register (DDRPHYC_DBYTEx_RXFIFOVISIBILITY) 550
14.5.22 DDRPHYC Rx FIFO content DQ321x register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ3210) 551

14.5.23 DDRPHYC Rx FIFO content DQ7654 register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ7654) 552
14.5.24 DDRPHYC Rx FIFO content DBI register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDBI) 552
14.5.25 DDRPHYC Tx slew rate controls register (DDRPHYC_DBYTEx_TXSLEWRATE_By) 553
14.5.26 DDRPHYC RxPBDlyTgx_r0 register (DDRPHYC_DBYTEx_RXPBDLYTG0_Ry) 553
14.5.27 DDRPHYC RxPBDlyTg1_rx register (DDRPHYC_DBYTEx_RXPBDLYTG1_Ry) 554
14.5.28 DDRPHYC RxEnDlyTgx_u0 register (DDRPHYC_DBYTEx_RXENDLYTG0_U0) 554
14.5.29 DDRPHYC RxEnDlyTg1_ux register (DDRPHYC_DBYTEx_RXENDLYTG1_U0) 555
14.5.30 DDRPHYC RxClkDlyTgx_u0 register (DDRPHYC_DBYTEx_RXCLKDLYTG0_U0) 555
14.5.31 DDRPHYC RxClkDlyTg1_ux register (DDRPHYC_DBYTEx_RXCLKDLYTG1_U0) 556
14.5.32 DDRPHYC RxClkcDlyTgx_u0 register (DDRPHYC_DBYTEx_RXCLKCDLYTG0_U0) 556
14.5.33 DDRPHYC RxClkcDlyTg1_ux register (DDRPHYC_DBYTEx_RXCLKCDLYTG1_U0) 557
14.5.34 DDRPHYC DqLnSel register (DDRPHYC_DBYTEx_DQyLNSEL) . . . 557
14.5.35 DDRPHYC TxDqDlyTgx_r0 register (DDRPHYC_DBYTEx_TXDQDLYTG0_Ry) 558
14.5.36 DDRPHYC TxDqDlyTg1_rx register (DDRPHYC_DBYTEx_TXDQDLYTG1_Ry) 558
14.5.37 DDRPHYC TxDqsDlyTgx_u0 register (DDRPHYC_DBYTEx_TXDQSDLYTG0_U0) 559
14.5.38 DDRPHYC TxDqsDlyTg1_ux register (DDRPHYC_DBYTEx_TXDQSDLYTG1_U0) 559
14.5.39 DDRPHYC debug status of the DBYTE LCDL register (DDRPHYC_DBYTEx_DXLCDLSTATUS) 560
14.5.40 DDRPHYC RxEnDlyTgx_u1 register (DDRPHYC_DBYTEx_RXENDLYTG0_U1) 561
14.5.41 DDRPHYC RxEnDlyTg1_u1 register (DDRPHYC_DBYTEx_RXENDLYTG1_U1) 561
14.5.42 DDRPHYC RxClkDlyTgx_u1 register (DDRPHYC_DBYTEx_RXCLKDLYTG0_U1) 562
14.5.43 DDRPHYC RxClkDlyTg1_u1 register (DDRPHYC_DBYTEx_RXCLKDLYTG1_U1) 562
14.5.44 DDRPHYC RxClkcDlyTgx_u1 register (DDRPHYC_DBYTEx_RXCLKCDLYTG0_U1) 563

14.5.45 DDRPHYC RxClkcDlyTg1_u1 register (DDRPHYC_DBYTEx_RXCLKCDLYTG1_U1) 563
14.5.46 DDRPHYC TxDqsDlyTgx_u1 register (DDRPHYC_DBYTEx_TXDQSDLYTG0_U1) 564
14.5.47 DDRPHYC TxDqsDlyTg1_u1 register (DDRPHYC_DBYTEx_TXDQSDLYTG1_U1) 564
14.5.48 DDRPHYC Rx FIFO pointer initialization control register (DDRPHYC_MASTER0_RXFIFOINIT) 565
14.5.49 DDRPHYC ForceClkDisable register (DDRPHYC_MASTER0_FORCECLKDISABLE) 565
14.5.50 DDRPHYC ForceInternalUpdate register (DDRPHYC_MASTER0_FORCEINTERNALUPDATE) 566
14.5.51 DDRPHYC read-only displays PHY configuration register (DDRPHYC_MASTER0_PHYCONFIG) 566
14.5.52 DDRPHYC PHY general configuration register (DDRPHYC_MASTER0_PGCR) 567
14.5.53 DDRPHYC test bump control1 register (DDRPHYC_MASTER0_TESTBUMPCNTRL1) 568
14.5.54 DDRPHYC impedance calibration clock ratio register (DDRPHYC_MASTER0_CALUCLKINFO) 568
14.5.55 DDRPHYC test bump control register (DDRPHYC_MASTER0_TESTBUMPCNTRL) 569
14.5.56 DDRPHYC Seq0BDLY0 register (DDRPHYC_MASTER0_SEQ0BDLY0) 570
14.5.57 DDRPHYC Seq0BDLY1 register (DDRPHYC_MASTER0_SEQ0BDLY1) 570
14.5.58 DDRPHYC Seq0BDLY2 register (DDRPHYC_MASTER0_SEQ0BDLY2) 571
14.5.59 DDRPHYC Seq0BDLY3 register (DDRPHYC_MASTER0_SEQ0BDLY3) 572
14.5.60 DDRPHYC PHY alert status bit register (DDRPHYC_MASTER0_PHYALERTSTATUS) 573
14.5.61 DDRPHYC setup intervals for DFI PHY master operation register (DDRPHYC_MASTER0_PPTTRAINSETUP) 574
14.5.62 DDRPHYC ATestMode control register (DDRPHYC_MASTER0_ATESTMODE) 575
14.5.63 DDRPHYC TxCalBinP register (DDRPHYC_MASTER0_TXCALBINP) 575
14.5.64 DDRPHYC TxCalBinN register (DDRPHYC_MASTER0_TXCALBINN) 576
14.5.65 DDRPHYC TX P impedance calibration override register (DDRPHYC_MASTER0_TXCALPOVR) 576

14.5.66 DDRPHYC TX N impedance calibration override register (DDRPHYC_MASTER0_TXCALNOVR) 577
14.5.67 DDRPHYC enables for update and low-power interfaces for DFI0 and DFI1 register (DDRPHYC_MASTER0_DFIMODE) 578
14.5.68 DDRPHYC mode select register for MEMCLK/address/command tristate register (DDRPHYC_MASTER0_TRISTATEMODECA) 579
14.5.69 DDRPHYC MtestMuxSel register (DDRPHYC_MASTER0_MTESTMUXSEL) 580
14.5.70 DDRPHYC MtestPgmInfo register (DDRPHYC_MASTER0_MTESTPGMINFO) 580
14.5.71 DDRPHYC dynamic power up/down control register (DDRPHYC_MASTER0_DYNPWRDNUP) 581
14.5.72 DDRPHYC PMIEnable register (DDRPHYC_MASTER0_PMIENABLE) 581
14.5.73 DDRPHYC PhyTID register (DDRPHYC_MASTER0_PHYTID) 582
14.5.74 DDRPHYC HwtMRL register (DDRPHYC_MASTER0_HWTMRL) . . . 582
14.5.75 DDRPHYC DFI PhyUpdate request time counter register (DDRPHYC_MASTER0_DFIPHYUPD) 583
14.5.76 DDRPHYC PdaMrsWriteMode register (DDRPHYC_MASTER0_PDAMRSWRITEMODE) 584
14.5.77 DDRPHYC DFIGEARDOWNCTL register (DDRPHYC_MASTER0_DFIGEARDOWNCTL) 585
14.5.78 DDRPHYC control the PHY logic related to the read and write DQS preamble register (DDRPHYC_MASTER0_DQSPREAMBLECONTROL) 585
14.5.79 DDRPHYC MasterX4Config register (DDRPHYC_MASTER0_MASTERX4CONFIG) 586
14.5.80 DDRPHYC write level feedback DQ observability selection register (DDRPHYC_MASTER0_WRLEVBITS) 587
14.5.81 DDRPHYC EnableCsMulticast register (DDRPHYC_MASTER0_ENABLECSMULTICAST) 588
14.5.82 DDRPHYC Acx4AnibDis register (DDRPHYC_MASTER0_ACX4ANIBDIS) 588
14.5.83 DDRPHYC enable Read-DBI function in each DBYTE register (DDRPHYC_MASTER0_DMIPINPRESENT) 589
14.5.84 DDRPHYC ARdPtrInitVal register (DDRPHYC_MASTER0_ARDPTRINITVAL) 589
14.5.85 DDRPHYC DLL mode control for DBYTEs register (DDRPHYC_MASTER0_DBYTEDLLMODECNTRL) 590
14.5.86 IDDRPHYC impedance calibration offsets control register (DDRPHYC_MASTER0_CALOFFSETS) 590
14.5.87 DDRPHYC SarInitVals register (DDRPHYC_MASTER0_SARINITVALS) 591

14.5.88 DDRPHYC CalPExtOvr register (DDRPHYC_MASTER0_CALPEXTOVR) 591
14.5.89 DDRPHYC CalCmpr5Ovr register (DDRPHYC_MASTER0_CALCMPR5OVR) 592
14.5.90 DDRPHYC CalNIntOvr register (DDRPHYC_MASTER0_CALNINTOVR) 592
14.5.91 DDRPHYC impedance calibration driver strength control register (DDRPHYC_MASTER0_CALDRVSTR0) 593
14.5.92 DDRPHYC read data on-die termination timing control register (DDRPHYC_MASTER0_PROCODTTIMECTL) 593
14.5.93 DDRPHYC MemAlert receiver configuration register (DDRPHYC_MASTER0_MEMALERTCONTROL) 595
14.5.94 DDRPHYC MemAlert receiver configuration register 2 (DDRPHYC_MASTER0_MEMALERTCONTROL2) 596
14.5.95 DDRPHYC protection and control of BP_MemReset_L register (DDRPHYC_MASTER0_MEMRESETL) 597
14.5.96 DDRPHYC PUBMODE - HWT Mux select register (DDRPHYC_MASTER0_PUBMODE) 598
14.5.97 DDRPHYC misc PHY status bits register (DDRPHYC_MASTER0_MISCPHYSTATUS) 598
14.5.98 DDRPHYC CoreLoopbackSel register (DDRPHYC_MASTER0_CORELOOPBACKSEL) 599
14.5.99 DDRPHYC DLL various training parameters register (DDRPHYC_MASTER0_DLLTRAINPARAM) 599
14.5.100 DDRPHYC HwtLpCsEnBypass register (DDRPHYC_MASTER0_HWTLPCSENBYPASS) 600
14.5.101 DDRPHYC DFI command/address mode register (DDRPHYC_MASTER0_DFICAMODE) 600
14.5.102 DDRPHYC DLL lock state machine control register (DDRPHYC_MASTER0_DLLCONTROL) 601
14.5.103 DDRPHYC DLL update phase control register (DDRPHYC_MASTER0_PULSEDLLUPDATEPHASE) 602
14.5.104 DDRPHYC DLL gain control register (DDRPHYC_MASTER0_DLLGAINCTL) 603
14.5.105 DDRPHYC impedance calibration control register (DDRPHYC_MASTER0_CALRATE) 604
14.5.106 DDRPHYC CalZap register (DDRPHYC_MASTER0_CALZAP) 605
14.5.107 DDRPHYC PState register (DDRPHYC_MASTER0_PSTATE) 605
14.5.108 DDRPHYC PLL output control register (DDRPHYC_MASTER0_PLLOUTGATECONTROL) 606
14.5.109 DDRPHYC PMU power-on reset control register (DDRPHYC_MASTER0_PORCONTROL) 606
14.5.110 DDRPHYC CalBusy register (DDRPHYC_MASTER0_CALBUSY) . . . 607

14.5.111 DDRPHYC misc impedance calibration control register (DDRPHYC_MASTER0_CALMISC2) 607
14.5.112 DDRPHYC controls for disabling the impedance calibration of certain target register (DDRPHYC_MASTER0_CALMISC) 609
14.5.113 DDRPHYC CalVRef register (DDRPHYC_MASTER0_CALVREFS) 610
14.5.114 DDRPHYC CalCmpr5 register (DDRPHYC_MASTER0_CALCMPR5) 610
14.5.115 DDRPHYC impedance calibration NInt control register (DDRPHYC_MASTER0_CALNINT) 611
14.5.116 DDRPHYC impedance calibration PExt control register (DDRPHYC_MASTER0_CALPEXT) 611
14.5.117 DDRPHYC impedance calibration Cmp invert control register (DDRPHYC_MASTER0_CALCMPINVERT) 612
14.5.118 DDRPHYC impedance calibration Cmpana control register (DDRPHYC_MASTER0_CALCMPANACNTRL) 612
14.5.119 DDRPHYC maps dfi_rddata_cs_n to destination dimm timing group register (DDRPHYC_MASTER0_DFIRDDATACSDESTMAP) . 613
14.5.120 PHY global V_{REF} controls register (DDRPHYC_MASTER0_VREFINGLOBAL) 613
14.5.121 DDRPHYC maps dfi_rddata_cs_n to destination dimm timing group register (DDRPHYC_MASTER0_DFIWRDATACSDESTMAP) . 614
14.5.122 DDRPHYC MasUpdGoodCtr register (DDRPHYC_MASTER0_MASUPDGOODCTR) 615
14.5.123 DDRPHYC PhyUpd0GoodCtr register (DDRPHYC_MASTER0_PHYUPD0GOODCTR) 616
14.5.124 DDRPHYC PhyUpd1GoodCtr register (DDRPHYC_MASTER0_PHYUPD1GOODCTR) 616
14.5.125 DDRPHYC CtlUpd0GoodCtr register (DDRPHYC_MASTER0_CTLUPD0GOODCTR) 617
14.5.126 DDRPHYC MasUpdFailCtr register (DDRPHYC_MASTER0_MASUPDFAILCTR) 617
14.5.127 DDRPHYC PhyUpd0FailCtr register (DDRPHYC_MASTER0_PHYUPD0FAILCTR) 618
14.5.128 DDRPHYC performance counter enable register (DDRPHYC_MASTER0_PHYPERFCTRENABLE) 618
14.5.129 DDRPHYC PllPwrDn register (DDRPHYC_MASTER0_PLLPWRDN) 619
14.5.130 DDRPHYC PllReset register (DDRPHYC_MASTER0_PLLRESET) . . 619
14.5.131 DDRPHYC PState dependent PLL control 2 register (DDRPHYC_MASTER0_PLLCTRL2) 620
14.5.132 DDRPHYC PLL control 0 register (DDRPHYC_MASTER0_PLLCTRL0) 620

14.5.133 DDRPHYC PState dependent PLL control 1 register (DDRPHYC_MASTER0_PLLCTRL1) 621
14.5.134 DDRPHYC PLL testing control register (DDRPHYC_MASTER0_PLLTST) 622
14.5.135 DDRPHYC PllLockStatus register (DDRPHYC_MASTER0_PLLLOCKSTATUS) 623
14.5.136 DDRPHYC PII TestMode register (DDRPHYC_MASTER0_PLLTESTMODE) 624
14.5.137 DDRPHYC PLL control 3 register (DDRPHYC_MASTER0_PLLCTRL3) 624
14.5.138 DDRPHYC PState dependent PLL control 4 register (DDRPHYC_MASTER0_PLLCTRL4) 625
14.5.139 DDRPHYC PllEndofCal register (DDRPHYC_MASTER0_PLLENDOFCAL) 625
14.5.140 DDRPHYC PllStandbyEff register (DDRPHYC_MASTER0_PLLSTANDBYEFF) 626
14.5.141 DDRPHYC PllDacValOut register (DDRPHYC_MASTER0_PLLDACVALOUT) 626
14.5.142 DDRPHYC LcdlDbgCntl register (DDRPHYC_MASTER0_LCDLDBGCNTL) 627
14.5.143 DDRPHYC Debug status of the DBYTE LCDL register (DDRPHYC_MASTER0_ACLCDLSTATUS) 628
14.5.144 DDRPHYC CUSTPHYREV register (DDRPHYC_MASTER0_CUSTPHYREV) 628
14.5.145 DDRPHYC PHYREV register (DDRPHYC_MASTER0_PHYREV) . . . 629
14.5.146 DDRPHYC Start vector value for LP3-exit or init PIE sequence register (DDRPHYC_MASTER0_LP3EXITSEQ0BSTARTVECTOR) . 629
14.5.147 DDRPHYC DFI frequency translation 0 register (DDRPHYC_MASTER0_DFIFREQXLAT0) 630
14.5.148 DDRPHYC DFI frequency translation 1 register (DDRPHYC_MASTER0_DFIFREQXLAT1) 630
14.5.149 DDRPHYC DFI frequency translation 2 register (DDRPHYC_MASTER0_DFIFREQXLAT2) 631
14.5.150 DDRPHYC DFI frequency translation 3 register (DDRPHYC_MASTER0_DFIFREQXLAT3) 631
14.5.151 DDRPHYC DFI frequency translation 4 register (DDRPHYC_MASTER0_DFIFREQXLAT4) 632
14.5.152 DDRPHYC DFI frequency translation 5 register (DDRPHYC_MASTER0_DFIFREQXLAT5) 632
14.5.153 DDRPHYC DFI frequency translation 6 register (DDRPHYC_MASTER0_DFIFREQXLAT6) 633
14.5.154 DDRPHYC DFI frequency translation 7 register (DDRPHYC_MASTER0_DFIFREQXLAT7) 633

14.5.155 DDRPHYC TxRdPtrInit register (DDRPHYC_MASTER0_TXRDPTRINIT) 634
14.5.156 DDRPHYC DfiInitComplete register (DDRPHYC_MASTER0_DFIINITCOMPLETE) 634
14.5.157 DDRPHYC DfiFreqRatio register (DDRPHYC_MASTER0_DFIFREQRATIO) 635
14.5.158 DDRPHYC RxFifoCheck register (DDRPHYC_MASTER0_RXFIFOCHECKS) 635
14.5.159 DDRPHYC MTestCombo output enable on core-side copy register (DDRPHYC_MASTER0_MTESTDTOCTRL) 636
14.5.160 DDRPHYC MapCAA[0to7]toDfi register (DDRPHYC_MASTER0_MAPCAAxTODFI) 636
14.5.161 DDRPHYC interrupt enable register (DDRPHYC_MASTER0_PHYINTERRUPTENABLE) 637
14.5.162 DDRPHYC interrupt firmware control register (DDRPHYC_MASTER0_PHYINTERRUPTFWCONTROL) 638
14.5.163 DDRPHYC interrupt mask register (DDRPHYC_MASTER0_PHYINTERRUPTMASK) 638
14.5.164 DDRPHYC interrupt clear register (DDRPHYC_MASTER0_PHYINTERRUPTCLEAR) 639
14.5.165 DDRPHYC interrupt status register (DDRPHYC_MASTER0_PHYINTERRUPTSTATUS) 640
14.5.166 DDRPHYC HwtSwizzleHwtAddress0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESSx) 642
14.5.167 DDRPHYC HwtSwizzleHwtActN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTACTN) 644
14.5.168 DDRPHYC HwtSwizzleHwtBank0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK0) 644
14.5.169 DDRPHYC HwtSwizzleHwtBank1 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK1) 645
14.5.170 DDRPHYC HwtSwizzleHwtBank2 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK2) 645
14.5.171 DDRPHYC HwtSwizzleHwtBg0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBG0) 646
14.5.172 DDRPHYC HwtSwizzleHwtBg1 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBG1) 646
14.5.173 DDRPHYC HwtSwizzleHwtCasN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTCASN) 647
14.5.174 DDRPHYC HwtSwizzleHwtRasN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTRASN) 647
14.5.175 DDRPHYC HwtSwizzleHwtWeN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTWEN) 648

14.5.176 DDRPHYC HwtSwizzleHwtParityIn register (DDRPHYC_MASTER0_HWTSWIZZLEHWTPARITYIN) 648
14.5.177 DDRPHYC small delays on handshake signals register 0 (DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS0) 649
14.5.178 DDRPHYC small delays on handshake signals register 1 (DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS1) 649
14.5.179 DDRPHYC remote impedance calibration status register (DDRPHYC_MASTER0_REMOTEIMPCAL) 650
14.5.180 DDRPHYC PhyInLP3 register (DDRPHYC_INITENG0_PHYINLP3) . 651
14.5.181 DDRPHYC Ucclk and Hclk enable register (DDRPHYC_DRTUB0_UCCLKHCLKENABLES) 651
14.5.182 DDRPHYC CurPstate0b register (DDRPHYC_DRTUB0_CURPSTATE0B) 652
14.5.183 DDRPHYC CUSTPUBREV register (DDRPHYC_DRTUB0_CUSTPUBREV) 652
14.5.184 DDRPHYC PUBREV register (DDRPHYC_DRTUB0_PUBREV) 653
14.5.185 DDRPHYC MicroContMuxSel register (DDRPHYC_APBONLY0_MICROCONTMUXSEL) 653
14.5.186 DDRPHYC PMU/controller protocol, controller read-only shadow register (DDRPHYC_APBONLY0_UCTSHADOWREGS) . . . 654
14.5.187 DDRPHYC DctWriteOnly register (DDRPHYC_APBONLY0_DCTWRITEONLY) 654
14.5.188 DDRPHYC DctWriteProt register (DDRPHYC_APBONLY0_DCTWRITEPROT) 655
14.5.189 DDRPHYC UctWriteOnlyShadow register (DDRPHYC_APBONLY0_UCTWRITEONLYSHADOW) 655
14.5.190 DDRPHYC UctDatWriteOnlyShadow register (DDRPHYC_APBONLY0_UCTDATWRITEONLYSHADOW) 656
14.5.191 DDRPHYC DfiCfgRdDataValidTicks register (DDRPHYC_APBONLY0_DFICFGRDDATAVALIDTICKS) 656
14.5.192 DDRPHYC reset and clock shutdown control register (DDRPHYC_APBONLY0_MICRORESET) 657
14.5.193 DDRPHYC DfiInitCompleteShadow register (DDRPHYC_APBONLY0_DFIINITCOMPLETESHADOW) 658
14.5.194 DDRPHYC register map 658

15 DDR performance monitor (DDRPERFM) 694

15.1 DDRPERFM introduction 694
15.2 DDRPERFM main features 694
15.3 DDRPERFM block diagram 694
15.4 DDRPERFM functional description 695

15.4.1 DDRPERFM performance monitor events 696
15.4.2 DDRPERFM DFI command decoder 698
15.5 DDRPERFM registers 700
15.5.1 DDRPERFM control register (DDRPERFM_CTRL) 700
15.5.2 DDRPERFM interrupt mask register (DDRPERFM_IMSK) 700
15.5.3 DDRPERFM interrupt clear register (DDRPERFM_ICLR) 701
15.5.4 DDRPERFM counter clear register (DDRPERFM_CLR) 701
15.5.5 DDRPERFM configuration register 0 (DDRPERFM_CFG0) 702
15.5.6 DDRPERFM configuration register 1 (DDRPERFM_CFG1) 702
15.5.7 DDRPERFM configuration register 2 (DDRPERFM_CFG2) 703
15.5.8 DDRPERFM configuration register 3 (DDRPERFM_CFG3) 704
15.5.9 DDRPERFM configuration register 4 (DDRPERFM_CFG4) 705
15.5.10 DDRPERFM event counter enable register (DDRPERFM_CFG5) . . . 705
15.5.11 DDRPERFM DRAM information register (DDRPERFM_DRAMINF) . . 705
15.5.12 DDRPERFM event counter 0 register (DDRPERFM_EVCNT0) 706
15.5.13 DDRPERFM event counter 1 register (DDRPERFM_EVCNT1) 706
15.5.14 DDRPERFM event counter 2 register (DDRPERFM_EVCNT2) 706
15.5.15 DDRPERFM event counter 3 register (DDRPERFM_EVCNT3) 707
15.5.16 DDRPERFM event counter 4 register (DDRPERFM_EVCNT4) 707
15.5.17 DDRPERFM event counter 5 register (DDRPERFM_EVCNT5) 707
15.5.18 DDRPERFM event counter 6 register (DDRPERFM_EVCNT6) 708
15.5.19 DDRPERFM event counter 7 register (DDRPERFM_EVCNT7) 708
15.5.20 DDRPERFM time counter register (DDRPERFM_TCNT) 708
15.5.21 DDRPERFM status register (DDRPERFM_STATUS) 709
15.5.22 DDRPERFM version register (DDRPERFM_VERR) 710
15.5.23 DDRPERFM identification register (DDRPERFM_ID) 710
15.5.24 DDRPERFM magic ID register (DDRPERFM_SID) 710
15.5.25 DDRPERFM register map 711

16 SRAM configuration controller (RAMCFG) 713
16.1 RAMCFG introduction 713
16.2 RAMCFG main features 713
16.3 RAMCFG functional description 713
16.3.1 Internal SRAM features 713
16.3.2 Error code correction (RETRAM, BKPSRAM) 714
16.3.3 Cyclic redundancy check (RETRAM) 715
16.5 RAMCFG interrupts 717

16.6 RAMCFG registers 717
16.6.1 RAMCFG SYSRAM control register (RAMCFG_SYSRAMCR) 717
16.6.2 RAMCFG SYSRAM interrupt status register (RAMCFG_SYSRAMISR) 718
16.6.3 RAMCFG SYSRAM erase key register (RAMCFG_SYSRAMERKEYR) 718
16.6.4 RAMCFG SRAM1 control register (RAMCFG_SRAM1CR) 719
16.6.5 RAMCFG SRAM1 interrupt status register (RAMCFG_SRAM1ISR) . . 719
16.6.6 RAMCFG SRAM1 erase key register (RAMCFG_SRAM1ERKEYR) . 720
16.6.7 RAMCFG RETRAM control register (RAMCFG_RETRAMCR) 720
16.6.8 RAMCFG RETRAM interrupt enable register (RAMCFG_RETRAMIER) 721
16.6.9 RAMCFG RETRAM interrupt status register (RAMCFG_RETRAMISR) 722
16.6.10 RAMCFG RETRAM ECC single error address register (RAMCFG_RETRAMSEAR) 722
16.6.11 RAMCFG RETRAM ECC double error address register (RAMCFG_RETRAMDEAR) 723
16.6.12 RAMCFG RETRAM interrupt clear flag register (RAMCFG_RETRAMICR) 723
16.6.13 RAMCFG RETRAM ECC key register (RAMCFG_RETRAMECCKEYR) 724
16.6.14 RAMCFG RETRAM erase key register (RAMCFG_RETRAMERKEYR) 724
16.6.15 RAMCFG RETRAM CRC control register 1 (RAMCFG_RETRAMCCR1) 725
16.6.16 RAMCFG RETRAM CRC control register 2 (RAMCFG_RETRAMCCR2) 725
16.6.17 RAMCFG RETRAM CRC reference signature register (RAMCFG_RETRAMCRSR) 726
16.6.18 RAMCFG RETRAM CRC status register (RAMCFG_RETRAMCSR) . 726
16.6.19 RAMCFG RETRAM CRC calculated signature register (RAMCFG_RETRAMCCSR) 727
16.6.20 RAMCFG BKPSRAM control register (RAMCFG_BKPSRAMCR) . . . 727
16.6.21 RAMCFG BKPSRAM interrupt enable register (RAMCFG_BKPSRAMIER) 728
16.6.22 RAMCFG BKPSRAM interrupt status register (RAMCFG_BKPSRAMISR) 729
16.6.23 RAMCFG BKPSRAM ECC single error address register (RAMCFG_BKPSRAMEEAR) 729
16.6.24 RAMCFG BKPSRAM ECC double error address register (RAMCFG_BKPSRAMDEAR) 730

16.6.25 RAMCFG BKPSRAM interrupt clear flag register (RAMCFG_BKPSRAMICR) . . . 730
16.6.26 RAMCFG BKPSRAM ECC key register (RAMCFG_BKPSRAMECCKEYR) . . . 731
16.6.27 RAMCFG BKPSRAM erase key register (RAMCFG_BKPSRAMERKEYR) . . . 731
16.6.28 RAMCFG register map . . . 732

17 Instruction cache (ICACHE) . . . 735

17.1 ICACHE introduction . . . 735
17.2 ICACHE main features . . . 735
17.3 ICACHE implementation . . . 736
17.4 ICACHE functional description . . . 736
17.4.1 ICACHE block diagram . . . 737
17.4.2 ICACHE reset and clocks . . . 737
17.4.3 ICACHE TAG memory . . . 738
17.4.4 Direct-mapped ICACHE (1-way cache) . . . 739
17.4.5 ICACHE enable . . . 740
17.4.6 Cacheable and noncacheable traffic . . . 740
17.4.7 Address remapping . . . 741
17.4.8 Cacheable accesses . . . 743
17.4.9 Dual-master cache . . . 744
17.4.10 ICACHE security . . . 744
17.4.11 ICACHE maintenance . . . 744
17.4.12 ICACHE performance monitoring . . . 745
17.4.13 ICACHE boot . . . 745
17.5 ICACHE low-power modes . . . 745
17.6 ICACHE error management and interrupts . . . 746
17.7 ICACHE registers . . . 746
17.7.1 ICACHE control register (ICACHE_CR) . . . 746
17.7.2 ICACHE status register (ICACHE_SR) . . . 747
17.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 748
17.7.4 ICACHE flag clear register (ICACHE_FCR) . . . 748
17.7.5 ICACHE hit monitor register (ICACHE_HMONR) . . . 749
17.7.6 ICACHE miss monitor register (ICACHE_MMONR) . . . 749
17.7.7 ICACHE region x configuration register (ICACHE_CRRx) . . . 749
17.7.8 ICACHE hardware configuration register (ICACHE_HWCFGR) . . . 750

17.7.9 ICACHE version register (ICACHE_VERR) . . . 752
17.7.10 ICACHE identification register (ICACHE_IPIDR) . . . 752
17.7.11 ICACHE size identification register (ICACHE_SIDR) . . . 753
17.7.12 ICACHE register map . . . 754

18 Data cache (DCACHE) . . . 756

18.1 DCACHE introduction . . . 756
18.2 DCACHE main features . . . 756
18.3 DCACHE implementation . . . 757
18.4 DCACHE functional description . . . 757
18.4.1 DCACHE block diagram . . . 758
18.4.2 DCACHE reset and clocks . . . 758
18.4.3 DCACHE TAG memory . . . 759
18.4.4 DCACHE enable . . . 761
18.4.5 Cacheable and noncacheable traffic . . . 761
18.4.6 Cacheable accesses . . . 762
18.4.7 DCACHE security . . . 764
18.4.8 DCACHE maintenance . . . 764
18.4.9 DCACHE performance monitoring . . . 766
18.4.10 DCACHE boot . . . 766
18.5 DCACHE low-power modes . . . 767
18.6 DCACHE error management and interrupts . . . 767
18.7 DCACHE registers . . . 768
18.7.1 DCACHE control register (DCACHE_CR) . . . 768
18.7.2 DCACHE status register (DCACHE_SR) . . . 769
18.7.3 DCACHE interrupt enable register (DCACHE_IER) . . . 770
18.7.4 DCACHE flag clear register (DCACHE_FCR) . . . 771
18.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR) . . . 771
18.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR) . . . 772
18.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR) . . . 772
18.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR) . . . 772
18.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR) . . . 773
18.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR) . . . 773
18.7.11 DCACHE hardware configuration register (DCACHE_HWCFGR) . . . 773
18.7.12 DCACHE version register (DCACHE_VERR) . . . 774

18.7.13 DCACHE identification register (DCACHE_IPIDR) 775
18.7.14 DCACHE size identification register (DCACHE_SIDR) 775
18.7.15 DCACHE register map 775

19 Power control (PWR) 777
19.1 PWR main features 777
19.2 PWR block diagram 778
19.2.1 PWR pin overview 779
19.3 PWR power supplies 779
19.3.1 PWR CPU and CORE domains 782
19.3.2 PWR supply system startup sequence 783
19.3.3 PWR supply Run2 sequence 784
19.3.4 PWR supply Stop sequences 785
19.3.5 PWR supply LP/LPV-Stop sequences 787
19.3.6 PWR supply Standby sequence 789
19.3.7 PWR supply VBAT sequence from Standby 791
19.3.8 PWR V_{SW} domain 791
19.3.9 PWR backup RAM (BKPSRAM) 793
19.3.10 PWR retention RAM (RETRAM) 793
19.3.11 PWR backup domain access 793
19.3.12 PWR V_{DD} domain 794
19.3.13 PWR analog supply 794
19.4 PWR power supply supervision 795
19.4.1 PWR power-on reset (POR)/power-down reset (PDR) 795
19.4.2 PWR brownout reset (BOR) 796
19.4.3 PWR V_{DDCORE} OK reset 797
19.4.4 PWR V_{DDCPU} OK reset 798
19.4.5 PWR programmable voltage detector (PVD) 798
19.4.6 Peripheral voltage monitoring (PVM) 799
19.4.7 PWR V08CAP voltage thresholds 801
19.4.8 PWR temperature thresholds 802
19.5 PWR power management 802
19.5.1 PWR power modes 809
19.5.2 Low-power batch acquisition mode (LPBAM) 813
19.5.3 PWR system wake-up 814
19.6 PWR low-power modes 814

19.6.1 PWR slowing down CPU and/or bus matrix clocks . . . 815
19.6.2 PWR controlling peripheral clocks . . . 815
19.6.3 PWR entering low-power modes . . . 815
19.6.4 PWR exiting from low-power modes . . . 816
19.6.5 PWR CSleep mode . . . 817
19.6.6 PWR CStop mode . . . 818
19.6.7 PWR D1 DStop1 mode . . . 819
19.6.8 PWR D1 DStandby mode . . . 819
19.6.9 PWR Stop1 mode . . . 820
19.6.10 PWR Stop2 mode . . . 821
19.6.11 PWR LP-Stop1 and LPLV-Stop1 modes . . . 822
19.6.12 PWR LP-Stop2 and LPLV-Stop2 modes . . . 823
19.6.13 PWR Standby mode . . . 824
19.7 PWR VBAT operation . . . 826
19.7.1 PWR debug Stop mode . . . 826
19.7.2 PWR debug Standby mode . . . 826
19.8 PWR WKUP pins . . . 827
19.9 PWR_ON, PWR_CPU_ON and PWR_LP pins . . . 827
19.10 PWR interrupts . . . 828
19.11 PWR resource isolation framework (RIF) . . . 828
19.11.1 PWR RIF configuration . . . 829
19.12 PWR registers . . . 831
19.12.1 PWR control register 1 (PWR_CR1) . . . 833
19.12.2 PWR control register 2 (PWR_CR2) . . . 835
19.12.3 PWR control register 3 (PWR_CR3) . . . 836
19.12.4 PWR control register 7 (PWR_CR7) . . . 837
19.12.5 PWR control register 8 (PWR_CR8) . . . 838
19.12.6 PWR control register 9 (PWR_CR9) . . . 840
19.12.7 PWR control register 10 (PWR_CR10) . . . 841
19.12.8 PWR control register 11 (PWR_CR11) . . . 842
19.12.9 PWR backup domain control register (PWR_BDCR) . . . 842
19.12.10 PWR CPU1 control register (PWR_CPU1CR) . . . 843
19.12.11 PWR CPU2 control register (PWR_CPU2CR) . . . 845
19.12.12 PWR D1 control register (PWR_D1CR) . . . 846
19.12.13 PWR D2 control register (PWR_D2CR) . . . 847
19.12.14 PWR wake-up control register x (PWR_WKUPCRx) . . . 849

19.12.15 PWR resource secure configuration register (PWR_RSECCFGR) . . . 850
19.12.16 PWR resource privileged configuration register (PWR_RPRIVCFGR) . . . 851
19.12.17 PWR resource x CID configuration register (PWR_RxCIDCFGR) . . . 852
19.12.18 PWR WIO secure configuration register (PWR_WIOSECCFGR) 853
19.12.19 PWR WIO privileged configuration register (PWR_WIOPRIVCFGR) . 854
19.12.20 PWR WIO x CID configuration register (PWR_WIOxCIDCFGR) 855
19.12.21 PWR WIO x semaphore control register (PWR_WIOxSEMCR) 857
19.12.22 PWR CPU1 status register (PWR_CPU1D1SR) . . . 858
19.12.23 PWR CPU2 status register (PWR_CPU2D2SR) . . . 859
19.12.24 PWR VER register (PWR_VERR) . . . 860
19.12.25 PWR identifier register (PWR_IPIDR) . . . 860
19.12.26 PWR size identification register (PWR_SIDR) . . . 860
19.12.27 PWR register map . . . 861

20 Reset and clock control (RCC) . . . 865

20.1 RCC main features . . . 865
20.2 RCC block diagram . . . 866
20.2.1 RCC pin overview . . . 867
20.3 RCC functional description - Reset . . . 867
20.3.1 Power-on/off resets . . . 867
20.3.2 Application and system resets . . . 868
20.3.3 Local resets . . . 870
20.3.4 Resets generated by CPUs . . . 870
20.3.5 Watchdog resets . . . 870
20.3.6 Backup domain reset . . . 871
20.3.7 CoreSight debug reset . . . 872
20.3.8 Option-byte loading . . . 872
20.3.9 Peripheral reset . . . 872
20.3.10 Reset pulse control (RPCTL) . . . 873
20.3.11 Reset coverage summary . . . 874
20.3.12 Reset source identification . . . 876
20.3.13 Power-on and wake-up sequences . . . 878
20.4 RCC functional description - Clocks . . . 885
20.4.1 Clock naming convention . . . 888
20.4.2 Oscillator description . . . 888
20.4.3 Clock security system (CSS) . . . 895

20.4.4 Clock output generation (MCO1 and MCO2) . . . 897
20.4.5 PLL description . . . 898
20.4.6 PLL root clocks . . . 903
20.4.7 Root clocks . . . 904
20.4.8 Clock generation in Stop and Standby modes . . . 910
20.4.9 Peripheral clock distribution . . . 918
20.4.10 General clock concept overview . . . 936
20.4.11 Peripheral allocation inside RCC . . . 938
20.4.12 Peripheral clock gating control . . . 940
20.4.13 CPUx and interconnect clocks gating . . . 944
20.4.14 Mass storage interface clocking . . . 947
20.4.15 Ethernet interface clocking . . . 951
20.4.16 CA35SS clocking . . . 956
20.4.17 Debug and trace clocking . . . 957
20.4.18 Camera subsystem clocking . . . 958
20.4.19 Display subsystem clocking . . . 959
20.4.20 USB clocking . . . 961
20.4.21 Security clocking . . . 964
20.5 RCC interrupts . . . 965
20.6 RCC application information . . . 967
20.6.1 Clock frequency calculation . . . 967
20.6.2 Clock calibration . . . 968
20.6.3 Clock monitoring . . . 969
20.7 RCC RIF function . . . 970
20.7.1 RCC RIF configuration . . . 970
20.8 RCC registers . . . 973
20.8.1 RCC secure configuration register 0 (RCC_SECCFGR0) . . . 983
20.8.2 RCC secure configuration register 1 (RCC_SECCFGR1) . . . 983
20.8.3 RCC secure configuration register 2 (RCC_SECCFGR2) . . . 984
20.8.4 RCC secure configuration register 3 (RCC_SECCFGR3) . . . 984
20.8.5 RCC privileged configuration register 0 (RCC_PRIVCFGR0) . . . 985
20.8.6 RCC privileged configuration register 1 (RCC_PRIVCFGR1) . . . 986
20.8.7 RCC privileged configuration register 2 (RCC_PRIVCFGR2) . . . 986
20.8.8 RCC privileged configuration register 3 (RCC_PRIVCFGR3) . . . 987
20.8.9 RCC resource configuration lock register 0 (RCC_RCFGLOCKR0) . . 988
20.8.10 RCC resource configuration lock register 1 (RCC_RCFGLOCKR1) . . 989
20.8.11 RCC resource configuration lock register 2 (RCC_RCFGLOCKR2) . . 989

20.8.12 RCC resource configuration lock register 3 (RCC_RCFGLOCKR3) . . 990
20.8.13 RCC resource x CID configuration register (RCC_RxCIDCFGR) 990
20.8.14 RCC resource x semaphore control register (RCC_RxSEMCR) 991
20.8.15 RCC global reset control set register (RCC_GRSTCSETR) 992
20.8.16 RCC CPU1 reset control set register (RCC_C1RSTCSETR) 993
20.8.17 RCC CPU2 reset control set register (RCC_C2RSTCSETR) 993
20.8.18 RCC hardware reset status clear register (RCC_HWRSTSCLRR) . . . 994
20.8.19 RCC CPU1 hardware reset status clear register (RCC_C1HWRSTSCLRR) 996
20.8.20 RCC CPU2 hardware reset status clear register (RCC_C2HWRSTSCLRR) 997
20.8.21 RCC CPU1 boot reset status set register (RCC_C1BOOTRSTSSETR) 997
20.8.22 RCC CPU1 boot reset status clear register (RCC_C1BOOTRSTSCLRR) 999
20.8.23 RCC CPU2 boot reset status set register (RCC_C2BOOTRSTSSETR) 1002
20.8.24 RCC CPU2 boot reset status clear register (RCC_C2BOOTRSTSCLRR) 1004
20.8.25 RCC CPU1 stop request set register (RCC_C1SREQSETR) 1007
20.8.26 RCC CPU1 stop request clear register (RCC_C1SREQCLRR) 1008
20.8.27 RCC CPU boot control register (RCC_CPUBOOTCR) 1008
20.8.28 RCC standby boot control register (RCC_STBYBOOTCR) 1009
20.8.29 RCC legacy boot control register (RCC_LEGBOOTCR) 1010
20.8.30 RCC backup domain control register (RCC_BDCR) 1011
20.8.31 RCC reset duration control register (RCC_RDCR) 1013
20.8.32 RCC C1MS reset delay control register (RCC_C1MSRDCR) 1014
20.8.33 RCC PWR_LP delay control register (RCC_PWRLPDLYCR) 1015
20.8.34 RCC CPU1 clock source interrupt enable set register (RCC_C1CIESETR) 1016
20.8.35 RCC CPU1 clock source interrupt flag clear register (RCC_C1CIFCLRR) 1017
20.8.36 RCC CPU2 clock source interrupt enable set register (RCC_C2CIESETR) 1019
20.8.37 RCC CPU2 clock source interrupt flag clear register (RCC_C2CIFCLRR) 1021
20.8.38 RCC IWDG CPU1 clock freeze set register (RCC_IWDGC1FZSETR) 1023
20.8.39 RCC IWDG CPU1 clock freeze clear register (RCC_IWDGC1FZCLRR) 1024

20.8.40 RCC IWDG CPU1 configuration set register (RCC_IWDGC1CFGSETR) . . . 1024
20.8.41 RCC IWDG CPU1 configuration clear register (RCC_IWDGC1CFGCLRR) . . . 1026
20.8.42 RCC IWDG CPU2 clock freeze set register (RCC_IWDGC2FZSETR) . . . 1027
20.8.43 RCC IWDG CPU2 clock freeze clear register (RCC_IWDGC2FZCLRR) . . . 1027
20.8.44 RCC IWDG CPU2 configuration set register (RCC_IWDGC2CFGSETR) . . . 1028
20.8.45 RCC IWDG CPU2 configuration clear register (RCC_IWDGC2CFGCLRR) . . . 1029
20.8.46 RCC MCO1 configuration register (RCC_MCO1CFGR) . . . 1030
20.8.47 RCC MCO2 configuration register (RCC_MCO2CFGR) . . . 1031
20.8.48 RCC oscillator clock enable set register (RCC_OCENSETR) . . . 1031
20.8.49 RCC oscillator clock enable clear register (RCC_OCENCLRR) . . . 1033
20.8.50 RCC oscillator clock ready register (RCC_OCRDYR) . . . 1034
20.8.51 RCC HSI configuration register (RCC_HSICFGR) . . . 1035
20.8.52 RCC MSI configuration register (RCC_MSICFGR) . . . 1036
20.8.53 RCC LSI control register (RCC_LSICR) . . . 1037
20.8.54 RCC RTC clock divider register (RCC_RTCDIVR) . . . 1037
20.8.55 RCC APB1 clock divider register (RCC_APB1DIVR) . . . 1038
20.8.56 RCC APB2 clock divider register (RCC_APB2DIVR) . . . 1039
20.8.57 RCC APB3 clock divider register (RCC_APB3DIVR) . . . 1039
20.8.58 RCC APB4 clock divider register (RCC_APB4DIVR) . . . 1040
20.8.59 RCC APB5 clock divider register (RCC_APB5DIVR) . . . 1041
20.8.60 RCC APBDBG clock divider register (RCC_APBDBGDIVR) . . . 1042
20.8.61 RCC TIM group 1 prescaler register (RCC_TIMG1PRER) . . . 1043
20.8.62 RCC TIM group 2 prescaler register (RCC_TIMG2PRER) . . . 1043
20.8.63 RCC LSMCU clock divider register (RCC_LSMCUDIVR) . . . 1044
20.8.64 RCC DDRCTRL AXI port configuration register (RCC_DDRCPCFGR) . . . 1045
20.8.65 RCC DDRCTRL APB configuration register (RCC_DDRCAPBCFGR) . . . 1046
20.8.66 RCC DDRPHYC APB configuration register (RCC_DDRPHYCAPBCFGR) . . . 1047
20.8.67 RCC DDRPHYC configuration register (RCC_DDRPHYCCFGR) . . 1048
20.8.68 RCC DDR configuration register (RCC_DDRCFGR) . . . 1048
20.8.69 RCC DDR interface configuration register (RCC_DDRITFCFGR) . . 1049
20.8.70 RCC SYSRAM configuration register (RCC_SYSRAMCFGR) . . . 1050

20.8.71 RCC SRAM1 configuration register (RCC_SRAM1CFGR) 1051
20.8.72 RCC RETRAM configuration register (RCC_RETRAMCFGR) 1051
20.8.73 RCC BKPSRAM configuration register (RCC_BKPSRAMCFGR) . . . 1052
20.8.74 RCC OCTOSPI1 configuration register (RCC_OSPI1CFGR) 1053
20.8.75 RCC FMC configuration register (RCC_FMCCFGR) 1054
20.8.76 RCC debug configuration register (RCC_DBGCFGR) 1054
20.8.77 RCC STM configuration register (RCC_STMCFGR) 1055
20.8.78 RCC ETR configuration register (RCC_ETRCFGR) 1056
20.8.79 RCC GPIOx configuration register (RCC_GPIOxCFGR) (x = A to I) . 1057
20.8.80 RCC GPIOZ configuration register (RCC_GPIOZCFGR) 1058
20.8.81 RCC HPDMA1 configuration register (RCC_HPDMA1CFGR) 1059
20.8.82 RCC HPDMA2 configuration register (RCC_HPDMA2CFGR) 1059
20.8.83 RCC HPDMA3 configuration register (RCC_HPDMA3CFGR) 1060
20.8.84 RCC IPCC1 configuration register (RCC_IPCC1CFGR) 1061
20.8.85 RCC RTC configuration register (RCC_RTCCFGR) 1062
20.8.86 RCC SYSCPU1 configuration register (RCC_SYSCPU1CFGR) . . . 1062
20.8.87 RCC BSEC configuration register (RCC_BSECCFGR) 1063
20.8.88 RCC PLL2 configuration register 1 (RCC_PLL2CFGR1) 1064
20.8.89 RCC PLL2 configuration register 2 (RCC_PLL2CFGR2) 1065
20.8.90 RCC PLL2 configuration register 3 (RCC_PLL2CFGR3) 1066
20.8.91 RCC PLL2 configuration register 4 (RCC_PLL2CFGR4) 1067
20.8.92 RCC PLL2 configuration register 5 (RCC_PLL2CFGR5) 1067
20.8.93 RCC PLL2 configuration register 6 (RCC_PLL2CFGR6) 1068
20.8.94 RCC PLL2 configuration register 7 (RCC_PLL2CFGR7) 1068
20.8.95 RCC HSI frequency monitoring control register (RCC_HSIFMONCR) 1069
20.8.96 RCC HSI frequency value register (RCC_HSIFVALR) 1070
20.8.97 RCC MSI frequency monitoring control register (RCC_MSIFMONCR) 1070
20.8.98 RCC MSI frequency value register (RCC_MSIFVALR) 1071
20.8.99 RCC TIM1 configuration register (RCC_TIM1CFGR) 1072
20.8.100 RCC TIM2 configuration register (RCC_TIM2CFGR) 1072
20.8.101 RCC TIM3 configuration register (RCC_TIM3CFGR) 1073
20.8.102 RCC TIM4 configuration register (RCC_TIM4CFGR) 1074
20.8.103 RCC TIM5 configuration register (RCC_TIM5CFGR) 1074
20.8.104 RCC TIM6 configuration register (RCC_TIM6CFGR) 1075
20.8.105 RCC TIM7 configuration register (RCC_TIM7CFGR) 1076
20.8.106 RCC TIM8 configuration register (RCC_TIM8CFGR) 1076

20.8.107 RCC TIM10 configuration register (RCC_TIM10CFGR) 1077
20.8.108 RCC TIM11 configuration register (RCC_TIM11CFGR) 1078
20.8.109 RCC TIM12 configuration register (RCC_TIM12CFGR) 1078
20.8.110 RCC TIM13 configuration register (RCC_TIM13CFGR) 1079
20.8.111 RCC TIM14 configuration register (RCC_TIM14CFGR) 1080
20.8.112 RCC TIM15 configuration register (RCC_TIM15CFGR) 1080
20.8.113 RCC TIM16 configuration register (RCC_TIM16CFGR) 1081
20.8.114 RCC TIM17 configuration register (RCC_TIM17CFGR) 1082
20.8.115 RCC LPTIM1 configuration register (RCC_LPTIM1CFGR) 1082
20.8.116 RCC LPTIM2 configuration register (RCC_LPTIM2CFGR) 1083
20.8.117 RCC LPTIM3 configuration register (RCC_LPTIM3CFGR) 1084
20.8.118 RCC LPTIM4 configuration register (RCC_LPTIM4CFGR) 1084
20.8.119 RCC LPTIM5 configuration register (RCC_LPTIM5CFGR) 1085
20.8.120 RCC SPI1 configuration register (RCC_SPI1CFGR) 1086
20.8.121 RCC SPI2 configuration register (RCC_SPI2CFGR) 1086
20.8.122 RCC SPI3 configuration register (RCC_SPI3CFGR) 1087
20.8.123 RCC SPI4 configuration register (RCC_SPI4CFGR) 1088
20.8.124 RCC SPI5 configuration register (RCC_SPI5CFGR) 1088
20.8.125 RCC SPI6 configuration register (RCC_SPI6CFGR) 1089
20.8.126 RCC SPDIFRX configuration register (RCC_SPDIFRXCFGR) 1090
20.8.127 RCC USART1 configuration register (RCC_USART1CFGR) 1090
20.8.128 RCC USART2 configuration register (RCC_USART2CFGR) 1091
20.8.129 RCC USART3 configuration register (RCC_USART3CFGR) 1092
20.8.130 RCC UART4 configuration register (RCC_UART4CFGR) 1092
20.8.131 RCC UART5 configuration register (RCC_UART5CFGR) 1093
20.8.132 RCC USART6 configuration register (RCC_USART6CFGR) 1094
20.8.133 RCC UART7 configuration register (RCC_UART7CFGR) 1094
20.8.134 RCC LPUART1 configuration register (RCC_LPUART1CFGR) 1095
20.8.135 RCC I2C1 configuration register (RCC_I2C1CFGR) 1096
20.8.136 RCC I2C2 configuration register (RCC_I2C2CFGR) 1096
20.8.137 RCC I2C3 configuration register (RCC_I2C3CFGR) 1097
20.8.138 RCC SAI1 configuration register (RCC_SAI1CFGR) 1098
20.8.139 RCC SAI2 configuration register (RCC_SAI2CFGR) 1098
20.8.140 RCC SAI3 configuration register (RCC_SAI3CFGR) 1099
20.8.141 RCC SAI4 configuration register (RCC_SAI4CFGR) 1100
20.8.142 RCC MDF1 configuration register (RCC_MDF1CFGR) 1100
20.8.143 RCC FDCAN configuration register (RCC_FDCANCFGR) 1101

20.8.144 RCC HDP configuration register (RCC_HDPCFGR) 1102
20.8.145 RCC ADC1 configuration register (RCC_ADC1CFGR) 1102
20.8.146 RCC ADC2 configuration register (RCC_ADC2CFGR) 1103
20.8.147 RCC Ethernet 1 configuration register (RCC_ETH1CFGR) 1104
20.8.148 RCC Ethernet 2 configuration register (RCC_ETH2CFGR) 1105
20.8.149 RCC USBH configuration register (RCC_USBHCFGR) 1107
20.8.150 RCC USB2PHY1 configuration register (RCC_USB2PHY1CFGR) . 1108
20.8.151 RCC OTG configuration register (RCC_OTGCFGR) 1109
20.8.152 RCC USB2PHY2 configuration register (RCC_USB2PHY2CFGR) . 1109
20.8.153 RCC STGEN configuration register (RCC_STGENCFGR) 1110
20.8.154 RCC SDMMC1 configuration register (RCC_SDMMC1CFGR) 1111
20.8.155 RCC SDMMC2 configuration register (RCC_SDMMC2CFGR) 1112
20.8.156 RCC SDMMC3 configuration register (RCC_SDMMC3CFGR) 1113
20.8.157 RCC LTDC configuration register (RCC_LTDCCFGR) 1114
20.8.158 RCC CSI configuration register (RCC_CSICFGR) 1114
20.8.159 RCC DCMIPP configuration register (RCC_DCMIPPCFGR) 1115
20.8.160 RCC DCMIPSSI configuration register (RCC_DCMIPSSICFGR) . . . 1116
20.8.161 RCC RNG1 configuration register (RCC_RNG1CFGR) 1116
20.8.162 RCC RNG2 configuration register (RCC_RNG2CFGR) 1117
20.8.163 RCC PKA configuration register (RCC_PKACFGR) 1118
20.8.164 RCC SAES configuration register (RCC_SAESCFGR) 1118
20.8.165 RCC HASH1 configuration register (RCC_HASH1CFGR) 1119
20.8.166 RCC HASH2 configuration register (RCC_HASH2CFGR) 1120
20.8.167 RCC CRYP1 configuration register (RCC_CRYP1CFGR) 1121
20.8.168 RCC CRYP2 configuration register (RCC_CRYP2CFGR) 1121
20.8.169 RCC IWDG1 configuration register (RCC_IWDG1CFGR) 1122
20.8.170 RCC IWDG2 configuration register (RCC_IWDG2CFGR) 1123
20.8.171 RCC IWDG3 configuration register (RCC_IWDG3CFGR) 1123
20.8.172 RCC IWDG4 configuration register (RCC_IWDG4CFGR) 1124
20.8.173 RCC WWDG1 configuration register (RCC_WWDG1CFGR) 1125
20.8.174 RCC VREFBUF configuration register (RCC_VREFCFGR) 1125
20.8.175 RCC DTS configuration register (RCC_DTSCFGR) 1126
20.8.176 RCC CRC configuration register (RCC_CRCCFGR) 1127
20.8.177 RCC SERC configuration register (RCC_SERCCFGR) 1128
20.8.178 RCC DDRPERFM configuration register (RCC_DDRPERFMCFGR) 1128
20.8.179 RCC I3C1 configuration register (RCC_I3C1CFGR) 1129
20.8.180 RCC I3C2 configuration register (RCC_I3C2CFGR) 1130

20.8.181 RCC I3C3 configuration register (RCC_I3C3CFGR) 1130
20.8.182 RCC MUXSEL configuration register (RCC_MUXSELCFGR) 1131
20.8.183 RCC cross bar x configuration register (RCC_XBARxCFGR) 1133
20.8.184 RCC pre divider x configuration register (RCC_PREDIVxCFGR) . . . 1134
20.8.185 RCC pre divider status register 1 (RCC_PREDIVSR1) 1134
20.8.186 RCC pre divider status register 2 (RCC_PREDIVSR2) 1135
20.8.187 RCC final divider x configuration register (RCC_FINDIVxCFGR) . . . 1135
20.8.188 RCC final divider status register 1 (RCC_FINDIVSR1) 1136
20.8.189 RCC final divider status register 2 (RCC_FINDIVSR2) 1136
20.8.190 RCC clock frequency calculator and observation 0 clock configuration register (RCC_FCALCOBS0CFGR) 1137
20.8.191 RCC clock frequency calculator and observation 1 clock configuration register (RCC_FCALCOBS1CFGR) 1139
20.8.192 RCC clock frequency calculator reference clock configuration register (RCC_FCALCREFCFGR) 1141
20.8.193 RCC clock frequency calculator control register 1 (RCC_FCALCCR1) 1142
20.8.194 RCC clock frequency calculator control register 2 (RCC_FCALCCR2) 1143
20.8.195 RCC clock frequency calculator status register (RCC_FCALCSR) . . 1144
20.8.196 RCC PLLy configuration register 1 (RCC_PLLyCFGR1) 1144
20.8.197 RCC PLLy configuration register 2 (RCC_PLLyCFGR2) 1145
20.8.198 RCC PLLy configuration register 3 (RCC_PLLyCFGR3) 1146
20.8.199 RCC PLLy configuration register 4 (RCC_PLLyCFGR4) 1147
20.8.200 RCC PLLy configuration register 5 (RCC_PLLyCFGR5) 1148
20.8.201 RCC PLLy configuration register 6 (RCC_PLLyCFGR6) 1148
20.8.202 RCC PLLy configuration register 7 (RCC_PLLyCFGR7) 1149
20.8.203 RCC version register (RCC_VERR) 1150
20.8.204 RCC identifier register (RCC_IDR) 1150
20.8.205 RCC size identification register (RCC_SIDR) 1150
20.8.206 RCC register map 1151

21 Inter-processor communication controller (IPCC) 1170

21.1 Introduction 1170
21.2 IPCC main features 1170
21.3 IPCC functional description 1171
21.3.1 IPCC block diagram 1171
21.3.2 IPCC Simplex channel mode 1172

21.3.3 IPCC Half-duplex channel mode 1175
21.3.4 IPCC interrupts and events 1178
21.3.5 IPCC security 1178
21.4 IPCC registers 1180
21.4.1 IPCC processor 1 control register (IPCC_C1CR) 1180
21.4.2 IPCC processor 1 mask register (IPCC_C1MR) 1181
21.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR) 1182
21.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR) 1182
21.4.5 IPCC processor 2 control register (IPCC_C2CR) 1183
21.4.6 IPCC processor 2 mask register (IPCC_C2MR) 1184
21.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR) 1185
21.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR) 1186
21.4.9 IPCC processor 1 security configuration register (IPCC_C1SECCFGR) 1187
21.4.10 IPCC processor 1 privilege configuration register (IPCC_C1PRIVCFGR) 1187
21.4.11 IPCC processor 1 CID configuration register (IPCC_C1CIDCFGR) 1188
21.4.12 IPCC processor 2 security configuration register (IPCC_C2SECCFGR) 1189
21.4.13 IPCC processor 2 privilege configuration register (IPCC_C2PRIVCFGR) 1190
21.4.14 IPCC processor 2 CID configuration register (IPCC_C2CIDCFGR) . 1190
21.4.15 IPCC hardware configuration register (IPCC_HWCFGR) 1191
21.4.16 IPCC version register (IPCC_VERR) 1192
21.4.17 IPCC identification register (IPCC_IPIDR) 1192
21.4.18 IPCC size identification register (IPCC_SIDR) 1192
21.4.19 IPCC register map 1193

22 General-purpose I/Os (GPIO) 1195

22.1 GPIO introduction 1195
22.2 GPIO main features 1195
22.3 GPIO functional description 1195
22.3.1 General-purpose I/O (GPIO) 1197
22.3.2 I/O pin alternate-function multiplexer and mapping 1197
22.3.3 I/O port control registers 1198
22.3.4 I/O port data registers 1198

22.3.5 I/O data bitwise handling . . . 1198
22.3.6 GPIO locking mechanism . . . 1199
22.3.7 I/O alternate-function input/output . . . 1199
22.3.8 External interrupt/wake-up lines . . . 1199
22.3.9 Input configuration . . . 1200
22.3.10 Output configuration . . . 1200
22.3.11 Alternate-function configuration . . . 1201
22.3.12 Analog configuration . . . 1201
22.3.13 GPIO pins used in V_{SW} domain . . . 1202
22.3.14 Advanced I/O configurations . . . 1202
22.3.15 I/O pin isolation using TrustZone . . . 1203
22.3.16 I/O pin isolation using privilege . . . 1204
22.3.17 I/O pin isolation using compartments . . . 1205
22.4 GPIO registers . . . 1206
22.4.1 GPIO port x mode register (GPIOx_MODER) (x = A to I, Z) . . . 1206
22.4.2 GPIO port x output type register (GPIOx_OTYPER) (x = A to I, Z) . . 1207
22.4.3 GPIO port x output speed register (GPIOx_OSPEEDR) (x = A to I, Z) . . . 1207
22.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I, Z) . . . 1208
22.4.5 GPIO port x input data register (GPIOx_IDR) (x = A to I, Z) . . . 1209
22.4.6 GPIO port x output data register (GPIOx_ODR) (x = A to I, Z) . . . 1209
22.4.7 GPIO port x bit set/reset register (GPIOx_BSRR) (x = A to I, Z) . . . 1210
22.4.8 GPIO port x configuration lock register (GPIOx_LCKR) (x = A to I, Z) . . . 1211
22.4.9 GPIO port x alternate function low register (GPIOx_AFRL) (x = A to I, Z) . . . 1212
22.4.10 GPIO port x alternate function high register (GPIOx_AFRH) (x = A to I, Z) . . . 1213
22.4.11 GPIO port x bit reset register (GPIOx_BRR) (x = A to I, Z) . . . 1214
22.4.12 GPIO port x secure configuration register (GPIOx_SECCFGR) (x = A to I, Z) . . . 1215
22.4.13 GPIO port x privileged configuration register (GPIOx_PRIVCFGR) (x = A to I, Z) . . . 1215
22.4.14 GPIO port x resource configuration lock register (GPIOx_RCFGLOCKR) (x = A to I, Z) . . . 1216
22.4.15 GPIO port x delay low register (GPIOx_DELAYRL) (x = A to I, Z) . . . 1216
22.4.16 GPIO port x delay high register (GPIOx_DELAYRH) (x = A to I, Z) . . 1217
22.4.17 GPIO port x advanced configuration low register (GPIOx_ADVCFGRL) (x = A to I, Z) . . . 1218

22.4.18 GPIO port x advanced configuration high register (GPIOx_ADVCFGRH) (x = A to I, Z) 1219
22.4.19 GPIO port x CID configuration register for I/O pin y (GPIOx_CIDCFGRy) (x = A to I, Z) 1221
22.4.20 GPIO port x semaphore control register for I/O pin y (GPIOx_SEMCRy) (x = A to I, Z) 1222
22.4.21 GPIO port x hardware configuration register 11 (GPIOx_HWCFGR11) (x = A to I, Z) 1223
22.4.22 GPIO port x hardware configuration register 10 (GPIOx_HWCFGR10) (x = A to I, Z) 1224
22.4.23 GPIO port x hardware configuration register 9 (GPIOx_HWCFGR9) (x = A to I, Z) 1224
22.4.24 GPIO port x hardware configuration register 8 (GPIOx_HWCFGR8) (x = A to I, Z) 1225
22.4.25 GPIO port x hardware configuration register 7 (GPIOx_HWCFGR7) (x = A to I, Z) 1225
22.4.26 GPIO port x hardware configuration register 6 (GPIOx_HWCFGR6) (x = A to I, Z) 1226
22.4.27 GPIO port x hardware configuration register 5 (GPIOx_HWCFGR5) (x = A to I, Z) 1226
22.4.28 GPIO port x hardware configuration register 4 (GPIOx_HWCFGR4) (x = A to I, Z) 1226
22.4.29 GPIO port x hardware configuration register 3 (GPIOx_HWCFGR3) (x = A to I, Z) 1227
22.4.30 GPIO port x hardware configuration register 2 (GPIOx_HWCFGR2) (x = A to I, Z) 1227
22.4.31 GPIO port x hardware configuration register 1 (GPIOx_HWCFGR1) (x = A to I, Z) 1227
22.4.32 GPIO port x version register (GPIOx_VERR) (x = A to I, Z) 1228
22.4.33 GPIO port x identification register (GPIOx_IPIDR) (x = A to I, Z) 1228
22.4.34 GPIO port x size identification register (GPIOx_SIDR) (x = A to I, Z) 1228
22.4.35 GPIO register map 1229

23 System configuration controller (SYSCFG) 1232

23.1 SYSCFG functional description 1232
23.1.1 I/O compensation cell 1232
23.2 SYSCFG registers 1233
23.2.1 SYSCFG boot pin status register (SYSCFG_BOOTSR) 1233
23.2.2 SYSCFG boot pin control register (SYSCFG_BOOTCR) 1234
23.2.3 SYSCFG DLYBSD1 control register (SYSCFG_DLYBSD1CR) 1235

23.2.4 SYSCFG DLYBSD1 status register (SYSCFG_DLYBSD1SR) 1236
23.2.5 SYSCFG SDMMC1 control register (SYSCFG_SDMMC1CR) 1236
23.2.6 SYSCFG SDMMC1 address remapping control register (SYSCFG_SDMMC1ARCR) 1237
23.2.7 SYSCFG DLYBSD2 control register (SYSCFG_DLYBSD2CR) 1237
23.2.8 SYSCFG DLYBSD2 status register (SYSCFG_DLYBSD2SR) 1238
23.2.9 SYSCFG SDMMC2 control register (SYSCFG_SDMMC2CR) 1239
23.2.10 SYSCFG SDMMC2 address remapping control register (SYSCFG_SDMMC2ARCR) 1239
23.2.11 SYSCFG DLYBSD3 control register (SYSCFG_DLYBSD3CR) 1240
23.2.12 SYSCFG DLYBSD3 status register (SYSCFG_DLYBSD3SR) 1241
23.2.13 SYSCFG SDMMC3 address remapping control register (SYSCFG_SDMMC3ARCR) 1242
23.2.14 SYSCFG DLYBOS1 control register (SYSCFG_DLYBOS1CR) 1242
23.2.15 SYSCFG DLYBOS1 status register (SYSCFG_DLYBOS1SR) 1244
23.2.16 SYSCFG potential tamper reset control register (SYSCFG_POTTAMPRSTCR) 1244
23.2.17 SYSCFG M33SS control register (SYSCFG_M33SSCR) 1245
23.2.18 SYSCFG ICN QOS priority control register 1 (SYSCFG_ICNQPCR1) 1246
23.2.19 SYSCFG ICN QOS priority control register 2 (SYSCFG_ICNQPCR2) 1247
23.2.20 SYSCFG ICN early write response control register (SYSCFG_ICNEWRCR) 1248
23.2.21 SYSCFG ICN end-to-end bandwidth regulation control register (SYSCFG_ICNE2EBWRCR) 1249
23.2.22 SYSCFG safe reset control register (SYSCFG_SAFERSTCR) 1250
23.2.23 SYSCFG ICN ETH bandwidth limiter control register (SYSCFG_ICNETHBWLCR) 1250
23.2.24 SYSCFG ICN CPU1 bandwidth limiter control register (SYSCFG_ICNCPU1BWLCR) 1251
23.2.25 SYSCFG ICN LTDC bandwidth limiter control register (SYSCFG_ICNLTDCBWLCR) 1252
23.2.26 SYSCFG ICN DCMIPP bandwidth limiter control register (SYSCFG_ICNDCMIPPBWLCR) 1253
23.2.27 SYSCFG ICN HPDMA bandwidth limiter control register (SYSCFG_ICNHPDMABWLCR) 1254
23.2.28 SYSCFG HPDMA address remapping control register (SYSCFG_HPDMAARCR) 1255
23.2.29 SYSCFG USB2PHY1 control register (SYSCFG_USB2PHY1CR) 1256

23.2.30 SYSCFG USB2PHY1 battery charging control register (SYSCFG_USB2PHY1BCCR) 1257
23.2.31 SYSCFG USB2PHY1 battery charging status register (SYSCFG_USB2PHY1BCSR) 1258
23.2.32 SYSCFG USB2PHY1 trimming 1 control register (SYSCFG_USB2PHY1TRIM1CR) 1259
23.2.33 SYSCFG USB2PHY1 trimming 2 control register (SYSCFG_USB2PHY1TRIM2CR) 1262
23.2.34 SYSCFG USBH control register (SYSCFG_USBHCR) 1263
23.2.35 SYSCFG USBH address remapping control register (SYSCFG_USBHARCR) 1264
23.2.36 SYSCFG USB2PHY2 control register (SYSCFG_USB2PHY2CR) 1264
23.2.37 SYSCFG USB2PHY2 trimming 1 control register (SYSCFG_USB2PHY2TRIM1CR) 1265
23.2.38 SYSCFG USB2PHY2 trimming 2 control register (SYSCFG_USB2PHY2TRIM2CR) 1269
23.2.39 SYSCFG OTG address remapping control register (SYSCFG_OTGARCR) 1270
23.2.40 SYSCFG Ethernet1 control register (SYSCFG_ETH1CR) 1270
23.2.41 SYSCFG Ethernet1 status register (SYSCFG_ETH1SR) 1271
23.2.42 SYSCFG Ethernet2 configuration register (SYSCFG_ETH2CR) 1271
23.2.43 SYSCFG Ethernet2 status register (SYSCFG_ETH2SR) 1272
23.2.44 SYSCFG VDDIO3 compensation cell control register (SYSCFG_VDDIO3CCCR) 1273
23.2.45 SYSCFG VDDIO3 compensation cell status register (SYSCFG_VDDIO3CCSR) 1274
23.2.46 SYSCFG VDD compensation cell control register (SYSCFG_VDDCCCR) 1274
23.2.47 SYSCFG VDD compensation cell status register (SYSCFG_VDDCCSR) 1275
23.2.48 SYSCFG VDDIO2 compensation cell control register (SYSCFG_VDDIO2CCCR) 1276
23.2.49 SYSCFG VDDIO2 compensation cell status register (SYSCFG_VDDIO2CCSR) 1277
23.2.50 SYSCFG VDDIO1 compensation cell control register (SYSCFG_VDDIO1CCCR) 1278
23.2.51 SYSCFG VDDIO1 compensation cell status register (SYSCFG_VDDIO1CCSR) 1279
23.2.52 SYSCFG control timer break register (SYSCFG_CBR) 1279
23.2.53 SYSCFG HDP USB debug control register (SYSCFG_HDPUSBDBGCR) 1280

23.2.54 SYSCFG device ID register (SYSCFG_IDC) 1281
23.2.55 SYSCFG version register (SYSCFG_VERR) 1281
23.2.56 SYSCFG identification register (SYSCFG_IPIDR) 1282
23.2.57 SYSCFG size identification register (SYSCFG_SIDR) 1282
23.2.58 SYSCFG register map 1282

24 Peripheral interconnects 1292
24.1 Introduction 1292
24.2 Connection overview 1292
24.2.1 Asynchronous connections (A) 1292
24.2.2 Synchronous connections (S) 1292
24.2.3 Break/fault connection for TIM outputs (B) 1292
24.2.4 Peripherals interconnect matrix 1293

25 High-performance direct memory access controller (HPDMA) . . . 1317
25.1 HPDMA introduction 1317
25.2 HPDMA main features 1317
25.3 HPDMA implementation 1318
25.3.1 HPDMA channels 1318
25.3.2 HPDMA allowed AXI maximum burst length 1319
25.3.3 HPDMA in low-power modes 1319
25.3.4 HPDMA requests 1320
25.3.5 HPDMA block requests 1324
25.3.6 HPDMA channels with peripheral early termination 1324
25.3.7 HPDMA triggers 1325
25.4 HPDMA functional description 1328
25.4.1 HPDMA block diagram 1328
25.4.2 HPDMA channel state and direct programming without any linked-list 1328
25.4.3 HPDMA channel suspend and resume 1329
25.4.4 HPDMA channel abort and restart 1330
25.4.5 HPDMA channel terminate and drain FIFO 1331
25.4.6 HPDMA linked-list data structure 1333
25.4.7 Linked-list item transfer execution 1336
25.4.8 HPDMA channel state and linked-list programming in run-to-completion mode 1337

25.4.9 HPDMA channel state and linked-list programming in link step mode . . . 1341
25.4.10 HPDMA channel state and linked-list programming . . . 1348
25.4.11 HPDMA FIFO-based transfers . . . 1350
25.4.12 HPDMA transfer request and arbitration . . . 1366
25.4.13 HPDMA triggered transfer . . . 1370
25.4.14 HPDMA circular buffering with linked-list programming . . . 1371
25.4.15 HPDMA transfer in peripheral flow-control mode . . . 1373
25.4.16 HPDMA byte-based padding at the block level . . . 1374
25.4.17 HPDMA secure/nonsecure channel . . . 1375
25.4.18 HPDMA privileged/unprivileged channel . . . 1376
25.4.19 HPDMA error management . . . 1377
25.5 HPDMA in debug mode . . . 1379
25.6 HPDMA in low-power modes . . . 1379
25.7 HPDMA interrupts . . . 1379
25.8 HPDMA registers . . . 1381
25.8.1 HPDMA secure configuration register (HPDMA_SECCFGR) . . . 1381
25.8.2 HPDMA privileged configuration register (HPDMA_PRIVCFGR) . . . 1382
25.8.3 HPDMA configuration lock register (HPDMA_RCFGLOCKR) . . . 1382
25.8.4 HPDMA nonsecure masked interrupt status register (HPDMA_MISR) . . . 1383
25.8.5 HPDMA secure masked interrupt status register (HPDMA_SMISR) . 1384
25.8.6 HPDMA channel x linked-list base address register (HPDMA_CxLBAR) . . . 1385
25.8.7 HPDMA channel x CID register (HPDMA_CxCIDCFGR) . . . 1385
25.8.8 HPDMA channel x semaphore control register (HPDMA_CxSEMCR) . . . 1387
25.8.9 HPDMA channel x flag clear register (HPDMA_CxFCR) . . . 1388
25.8.10 HPDMA channel x status register (HPDMA_CxSR) . . . 1389
25.8.11 HPDMA channel x control register (HPDMA_CxCR) . . . 1391
25.8.12 HPDMA channel x transfer register 1 (HPDMA_CxTR1) . . . 1394
25.8.13 HPDMA channel x transfer register 2 (HPDMA_CxTR2) . . . 1398
25.8.14 HPDMA channel x block register 1 (HPDMA_CxBR1) . . . 1402
25.8.15 HPDMA channel x alternate block register 1 (HPDMA_CxBR1) . . . 1403
25.8.16 HPDMA channel x source address register (HPDMA_CxSAR) . . . 1406
25.8.17 HPDMA channel x destination address register (HPDMA_CxDAR) . 1408
25.8.18 HPDMA channel x transfer register 3 (HPDMA_CxTR3) . . . 1410
25.8.19 HPDMA channel x block register 2 (HPDMA_CxBR2) . . . 1412

25.8.20 HPDMA channel x linked-list address register (HPDMA_CxLLR) . . . 1413
25.8.21 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR) . . . 1415
25.8.22 HPDMA hardware configuration 15 register (HPDMA_HWCFGR15) 1416
25.8.23 HPDMA hardware configuration 14 register (HPDMA_HWCFGR14) 1418
25.8.24 HPDMA hardware configuration 13 register (HPDMA_HWCFGR13) 1419
25.8.25 HPDMA hardware configuration 12 register (HPDMA_HWCFGR12) 1420
25.8.26 HPDMA hardware configuration 11 register (HPDMA_HWCFGR11) 1421
25.8.27 HPDMA hardware configuration 10 register (HPDMA_HWCFGR10) 1422
25.8.28 HPDMA hardware configuration 9 register (HPDMA_HWCFGR9) . . 1423
25.8.29 HPDMA hardware configuration 8 register (HPDMA_HWCFGR8) . . 1424
25.8.30 HPDMA hardware configuration 7 register (HPDMA_HWCFGR7) . . 1425
25.8.31 HPDMA hardware configuration 6 register (HPDMA_HWCFGR6) . . 1426
25.8.32 HPDMA hardware configuration 5 register (HPDMA_HWCFGR5) . . 1428
25.8.33 HPDMA hardware configuration 4 register (HPDMA_HWCFGR4) . . 1429
25.8.34 HPDMA hardware configuration 3 register (HPDMA_HWCFGR3) . . 1430
25.8.35 HPDMA hardware configuration 2 register (HPDMA_HWCFGR2) . . 1431
25.8.36 HPDMA hardware configuration 1 register (HPDMA_HWCFGR1) . . 1431
25.8.37 HPDMA version register (HPDMA_VERR) . . . 1433
25.8.38 HPDMA identification register (HPDMA_IPIDR) . . . 1433
25.8.39 HPDMA size identification register (HPDMA_SIDR) . . . 1433
25.8.40 HPDMA register map . . . 1434

26 Interrupt list . . . 1438
26.1 NVIC interrupts . . . 1438
26.2 GIC interrupts . . . 1447

27 Nested vectored interrupt controller (NVIC) . . . 1458
27.1 NVIC features . . . 1458
27.2 SysTick calibration value register . . . 1458
27.3 Interrupt and exception vectors . . . 1458

28 Global interrupt controller (GIC) . . . 1459
28.1 GIC introduction . . . 1459
28.2 GIC main features . . . 1459
28.3 GIC functional description . . . 1459
28.4 GIC interrupt sources . . . 1460

28.4.1 Software generated interrupts (SGI) 1460
28.4.2 Private peripheral interrupts (PPI) 1460
28.4.3 Shared peripheral interrupts (SPI) 1461
28.4.4 Interrupt priority formats 1461
28.5 GIC distributor (GICD) 1461
28.5.1 GICD control register (GICD_CTLR) 1462
28.5.2 GICD control non-secure access register (GICD_CTLRNS) 1462
28.5.3 GICD interrupt controller type register (GICD_TYPER) 1463
28.5.4 GICD implementer identification register (GICD_IIDR) 1463
28.5.5 GICD interrupt group register x (GICD_IGROUPRx) 1464
28.5.6 GICD interrupt set-enable register (GICD_ISENABLER0) 1464
28.5.7 GICD interrupt set-enable register x (GICD_ISENABLERx) 1464
28.5.8 GICD interrupt clear-enable register (GICD_ICENABLER0) 1465
28.5.9 GICD interrupt clear-enable register x (GICD_ICENABLERx) 1465
28.5.10 GICD interrupt set-pending register x (GICD_ISPENDRx) 1465
28.5.11 GICD interrupt clear-pending register x (GICD_ICPENDRx) 1466
28.5.12 GICD interrupt set-active register x (GICD_ISACTIVERx) 1466
28.5.13 GICD interrupt clear-active register x (GICD_ICACTIVERx) 1466
28.5.14 GICD interrupt priority register x (GICD_IPRIORITYRx) 1467
28.5.15 GICD interrupt processor target register x (GICD_ITARGETSRx) . . 1467
28.5.16 GICD interrupt processor target register x (GICD_ITARGETSRx) . . 1468
28.5.17 GICD interrupt configuration register (GICD_ICFGR0) 1468
28.5.18 GICD interrupt configuration register (GICD_ICFGR1) 1469
28.5.19 GICD interrupt configuration register x (GICD_ICFGRx) 1469
28.5.20 GICD private peripheral interrupt status register (GICD_PPISR) . . . 1469
28.5.21 GICD shared peripheral interrupt register x (GICD_SPISRx) 1470
28.5.22 GICD software generated interrupt register (GICD_SGIR) 1470
28.5.23 GICD SGI clear-pending register x (GICD_CPENDSGIRx) 1471
28.5.24 GICD SGI set-pending register x (GICD_SPENDSGIRx) 1472
28.5.25 GICD peripheral ID4 register (GICD_PIDR4) 1473
28.5.26 GICD peripheral ID5 to ID7 register x (GICD_PIDRx) 1473
28.5.27 GICD peripheral ID0 register (GICD_PIDR0) 1473
28.5.28 GICD peripheral ID1 register (GICD_PIDR1) 1474
28.5.29 GICD peripheral ID2 register (GICD_PIDR2) 1474
28.5.30 GICD peripheral ID3 register (GICD_PIDR3) 1474
28.5.31 GICD component ID0 register (GICD_CIDR0) 1475
28.5.32 GICD component ID1 register (GICD_CIDR1) 1475

28.5.33 GICD component ID2 register (GICD_CIDR2) 1475
28.5.34 GICD component ID3 register (GICD_CIDR3) 1476
28.5.35 GICD register map 1476
28.6 GIC CPU interface (GICC) 1479
28.6.1 GICC control register (GICC_CTLR) 1479
28.6.2 GICC control non-secure access register (GICC_CTLRNS) 1480
28.6.3 GICC input priority mask register (GICC_PMR) 1481
28.6.4 GICC binary point register (GICC_BPR) 1481
28.6.5 GICC binary point non-secure access register (GICC_BPRNS) 1482
28.6.6 GICC interrupt acknowledge register (GICC_IAR) 1482
28.6.7 GICC end of interrupt register (GICC_EOIR) 1483
28.6.8 GICC running priority register (GICC_RPR) 1483
28.6.9 GICC highest priority pending interrupt register (GICC_HPPIR) 1483
28.6.10 GICC aliased binary point register (GICC_ABPR) 1484
28.6.11 GICC aliased interrupt acknowledge register (GICC_AIAR) 1484
28.6.12 GICC aliased end of interrupt register (GICC_AEOIR) 1485
28.6.13 GICC aliased highest priority pending interrupt register (GICC_AHPPIR) 1485
28.6.14 GICC active priority register (GICC_APR0) 1486
28.6.15 GICC non-secure active priority register (GICC_NSAPR0) 1486
28.6.16 GICC interface identification register (GICC_IIDR) 1486
28.6.17 GICC deactivate interrupt register (GICC_DIR) 1487
28.6.18 GICC register map 1487
28.7 GIC virtual interface control, common (GICH) 1489
28.7.1 GICH hypervisor control register (GICH_HCR) 1489
28.7.2 GICH VGIC type register (GICH_VTR) 1490
28.7.3 GICH virtual machine control register (GICH_VMCR) 1490
28.7.4 GICH maintenance interrupt status register (GICH_MISR) 1491
28.7.5 GICH end of interrupt status register (GICH_EISR0) 1492
28.7.6 GICH empty list status register (GICH_ELSR0) 1492
28.7.7 GICH active priority register (GICH_APR0) 1492
28.7.8 GICH list register x (GICH_LRx) 1493
28.7.9 GICH register map 1493
28.8 GIC virtual CPU interface (GICV) 1494
28.8.1 GICV virtual machine control register (GICV_CTLR) 1495
28.8.2 GICV VM priority mask register (GICV_PMR) 1495
28.8.3 GICV VM binary point register (GICV_BPR) 1496

28.8.4 GICV VM interrupt acknowledge register (GICV_IAR) 1496
28.8.5 GICV VM end of interrupt register (GICV_EOIR) 1497
28.8.6 GICV VM running priority register (GICV_RPR) 1497
28.8.7 GICV VM highest priority pending interrupt register (GICV_HPPIR) . 1498
28.8.8 GICV VM aliased binary point register (GICV_ABPR) 1498
28.8.9 GICV VM aliased interrupt register (GICV_AIAR) 1499
28.8.10 GICV VM aliased end of interrupt register (GICV_AEOIR) 1499
28.8.11 GICV VM aliased highest priority pending interrupt register (GICV_AHPPIR) 1500
28.8.12 GICV VM active priority register (GICV_APR0) 1500
28.8.13 GICV VM CPU interface identification register (GICV_IIDR) 1501
28.8.14 GICV VM deactivate interrupt register (GICV_DIR) 1501
28.8.15 GICV register map 1501

29 Extended interrupts and event controller (EXTI) 1503

29.1 EXTI main features 1503
29.2 EXTI implementation 1504
29.3 EXTI block diagram 1504
29.3.1 EXTI connections between peripherals and CPU 1506
29.3.2 EXTI1 wake-up interrupt list 1507
29.3.3 EXTI2 wake-up interrupt list 1510
29.4 EXTI functional description 1512
29.4.1 EXTI configurable event input wake-up 1513
29.4.2 EXTI direct event input wake-up 1513
29.4.3 EXTI multiplexer selection 1515
29.5 EXTI functional behavior 1515
29.6 EXTI event protection 1516
29.6.1 EXTI register security protection 1517
29.6.2 EXTI register privilege protection 1517
29.6.3 EXTI register CID protection 1517
29.7 EXTIx registers 1518
29.7.1 EXTI1 rising trigger selection register (EXTI1_RTSR1) 1519
29.7.2 EXTI1 falling trigger selection register (EXTI1_FTSR1) 1520
29.7.3 EXTI1 software interrupt event register (EXTI1_SWIER1) 1520
29.7.4 EXTI1 rising edge pending register (EXTI1_RPR1) 1521
29.7.5 EXTI1 falling edge pending register (EXTI1_FPR1) 1522
29.7.6 EXTI1 security configuration register (EXTI1_SECCFGR1) 1522

29.7.7 EXTI1 privilege configuration register (EXTI1_PRIVCFGR1) 1523
29.7.8 EXTI1 rising trigger selection register (EXTI1_RTSR2) 1524
29.7.9 EXTI1 falling trigger selection register (EXTI1_FTSR2) 1524
29.7.10 EXTI1 software interrupt event register (EXTI1_SWIER2) 1525
29.7.11 EXTI1 rising edge pending register (EXTI1_RPR2) 1526
29.7.12 EXTI1 falling edge pending register (EXTI1_FPR2) 1527
29.7.13 EXTI1 security enable register (EXTI1_SECCFGR2) 1528
29.7.14 EXTI1 privilege enable register (EXTI1_PRIVCFGR2) 1529
29.7.15 EXTI1 rising trigger selection register (EXTI1_RTSR3) 1530
29.7.16 EXTI1 falling trigger selection register (EXTI1_FTSR3) 1531
29.7.17 EXTI1 software interrupt event register (EXTI1_SWIER3) 1532
29.7.18 EXTI1 rising edge pending register (EXTI1_RPR3) 1533
29.7.19 EXTI1 falling edge pending register (EXTI1_FPR3) 1534
29.7.20 EXTI1 security enable register (EXTI1_SECCFGR3) 1535
29.7.21 EXTI1 privilege enable register (EXTI1_PRIVCFGR3) 1536
29.7.22 EXTI1 external interrupt selection register 1 (EXTI1_EXTICR1) 1537
29.7.23 EXTI1 external interrupt selection register 2 (EXTI1_EXTICR2) 1539
29.7.24 EXTI1 external interrupt selection register 3 (EXTI1_EXTICR3) 1542
29.7.25 EXTI1 external interrupt selection register 4 (EXTI1_EXTICR4) 1545
29.7.26 EXTI1 lock register (EXTI1_LOCKR) 1548
29.7.27 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR1) (m = 1 to 2) 1549
29.7.28 EXTI1 CPU2 wake-up with event mask register (EXTI1_C2EMR1) . 1550
29.7.29 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR2) (m = 1 to 2) 1551
29.7.30 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR3) (m = 1 to 2) 1552
29.7.31 EXTI1 CPU2 wake-up with event mask register (EXTI1_C2EMR3) . 1553
29.7.32 EXTI1 event CID configuration register (EXTI1_EnCIDCFGR) 1554
29.7.33 EXTI1 processor m CID configuration register (EXTI1_CmCIDCFGR) 1554
29.7.34 EXTI1 hardware configuration registers 13 (EXTI1_HWCFGR13) . . 1555
29.7.35 EXTI1 hardware configuration registers 12 (EXTI1_HWCFGR12) . . 1555
29.7.36 EXTI1 hardware configuration registers 11 (EXTI1_HWCFGR11) . . 1556
29.7.37 EXTI1 hardware configuration registers 7 (EXTI1_HWCFGR7) 1556
29.7.38 EXTI1 hardware configuration registers 6 (EXTI1_HWCFGR6) 1556
29.7.39 EXTI1 hardware configuration registers 5 (EXTI1_HWCFGR5) 1557
29.7.40 EXTI1 hardware configuration registers 4 (EXTI1_HWCFGR4) 1557

29.7.41 EXTI1 hardware configuration registers 3 (EXTI1_HWCFGR3) 1557
29.7.42 EXTI1 hardware configuration registers 2 (EXTI1_HWCFGR2) 1558
29.7.43 EXTI1 hardware configuration register 1 (EXTI1_HWCFGR1) 1558
29.7.44 EXTI1 version register (EXTI1_VERR) 1558
29.7.45 EXTI1 identification register (EXTI1_IPIDR) 1559
29.7.46 EXTI1 size identification register (EXTI1_SIDR) 1559
29.7.47 EXTI2 rising trigger selection register (EXTI2_RTSR1) 1559
29.7.48 EXTI2 falling trigger selection register (EXTI2_FTSR1) 1560
29.7.49 EXTI2 software interrupt event register (EXTI2_SWIER1) 1561
29.7.50 EXTI2 rising edge pending register (EXTI2_RPR1) 1562
29.7.51 EXTI2 falling edge pending register (EXTI2_FPR1) 1563
29.7.52 EXTI2 security configuration register (EXTI2_SECCFGR1) 1563
29.7.53 EXTI2 privilege configuration register (EXTI2_PRIVCFGR1) 1564
29.7.54 EXTI2 rising trigger selection register (EXTI2_RTSR2) 1565
29.7.55 EXTI2 falling trigger selection register (EXTI2_FTSR2) 1566
29.7.56 EXTI2 software interrupt event register (EXTI2_SWIER2) 1567
29.7.57 EXTI2 rising edge pending register (EXTI2_RPR2) 1567
29.7.58 EXTI2 falling edge pending register (EXTI2_FPR2) 1568
29.7.59 EXTI2 security enable register (EXTI2_SECCFGR2) 1569
29.7.60 EXTI2 privilege enable register (EXTI2_PRIVCFGR2) 1570
29.7.61 EXTI2 external interrupt selection register 1 (EXTI2_EXTICR1) 1571
29.7.62 EXTI2 external interrupt selection register 2 (EXTI2_EXTICR2) 1573
29.7.63 EXTI2 external interrupt selection register 3 (EXTI2_EXTICR3) 1576
29.7.64 EXTI2 external interrupt selection register 4 (EXTI2_EXTICR4) 1578
29.7.65 EXTI2 lock register (EXTI2_LOCKR) 1581
29.7.66 EXTI2 CPUm wake-up with interrupt mask register (EXTI2_CmIMR1) (m = 1 to 2) 1581
29.7.67 EXTI2 CPU2 wake-up with event mask register (EXTI2_C2EMR1) 1582
29.7.68 EXTI2 CPUm wake-up with interrupt mask register (EXTI2_CmIMR2) (m = 1 to 2) 1583
29.7.69 EXTI2 CPU2 wake-up with event mask register (EXTI2_C2EMR2) 1584
29.7.70 EXTI2 event CID config register (EXTI2_EnCIDCFGR) 1584
29.7.71 EXTI2 processor m CID configuration register (EXTI2_CmCIDCFGR) 1585
29.7.72 EXTI2 hardware configuration registers 13 (EXTI2_HWCFGR13) . . 1586
29.7.73 EXTI2 hardware configuration registers 12 (EXTI2_HWCFGR12) . . 1586

29.7.74 EXTI2 hardware configuration registers 11 (EXTI2_HWCFGR11) . . 1586
29.7.75 EXTI2 hardware configuration registers 7 (EXTI2_HWCFGR7) 1587
29.7.76 EXTI2 hardware configuration registers 6 (EXTI2_HWCFGR6) 1587
29.7.77 EXTI2 hardware configuration registers 5 (EXTI2_HWCFGR5) 1587
29.7.78 EXTI2 hardware configuration registers 4 (EXTI2_HWCFGR4) 1588
29.7.79 EXTI2 hardware configuration registers 3 (EXTI2_HWCFGR3) 1588
29.7.80 EXTI2 hardware configuration registers 2 (EXTI2_HWCFGR2) 1588
29.7.81 EXTI2 hardware configuration register 1 (EXTI2_HWCFGR1) 1589
29.7.82 EXTI2 version register (EXTI2_VERR) 1589
29.7.83 EXTI2 identification register (EXTI2_IPIDR) 1589
29.7.84 EXTI2 size identification register (EXTI2_SIDR) 1590
29.7.85 EXTI1 register map 1590
29.7.86 EXTI2 register map 1595

30 Cyclic redundancy check calculation unit (CRC) 1599

30.1 CRC introduction 1599
30.2 CRC main features 1599
30.3 CRC functional description 1600
30.3.1 CRC block diagram 1600
30.3.2 CRC internal signals 1600
30.3.3 CRC operation 1600
30.4 CRC registers 1602
30.4.1 CRC data register (CRC_DR) 1602
30.4.2 CRC independent data register (CRC_IDR) 1602
30.4.3 CRC control register (CRC_CR) 1603
30.4.4 CRC initial value (CRC_INIT) 1604
30.4.5 CRC polynomial (CRC_POL) 1604
30.4.6 CRC register map 1605

31 Mass storage interfaces 1606

31.1 Overview 1606
31.2 FMC interface 1606
31.2.1 FMC interface block diagram 1607
31.2.2 FMC interface reset 1607
31.2.3 FMC interface interrupt 1608
31.3 Octo-SPI interface 1608

31.3.1 Main features of the Octo-SPI interface 1608
31.3.2 Octo-SPI interface block diagram 1609
31.3.3 Octo-SPI interface resets 1610
31.3.4 Octo-SPI interface memory 1610
31.3.5 OCTOSPI1 description 1611
31.3.6 OTFDEC1 description 1611
31.3.7 DLYBOS1 description 1611
31.3.8 Octo-SPI interface interrupts 1613
31.3.9 Chip select management 1613
31.3.10 Octo-SPI interface programmable parameters 1615
31.3.11 Select the RX sampling strobe and configure the TX clock 1616
31.4 SD/eMMC interface 1617
31.4.1 SD/eMMC interface main features 1618
31.4.2 SD/eMMC interface block diagram 1620
31.4.3 SD/eMMC interface reset 1621
31.4.4 SDMMC1/2 description 1621
31.4.5 DLYBSD1/2 description 1622
31.4.6 SD/eMMC interface interrupts 1622
31.4.7 SD/eMMC interface programmable parameters 1622
31.4.8 Guidelines for selecting the RX sampling clock 1624

32 Flexible memory controller (FMC) 1627
32.1 FMC main features 1627
32.2 FMC implementation 1628
32.3 FMC block diagram 1628
32.4 FMC internal signals 1629
32.5 AHB interface 1629
32.6 AXI interface 1630
32.6.1 Supported memories and transactions 1630
32.7 External device address mapping 1631
32.7.1 NOR/PSRAM address mapping 1631
32.7.2 NAND flash memory address mapping 1632
32.8 FMC security and isolation 1633
32.9 FMC RIF registers 1636
32.9.1 FMC security configuration register (FMC_SECCFGR) 1636
32.9.2 FMC security configuration register (FMC_PRIVCFGR) 1637

32.9.3 FMC CID configuration lock register (FMC_RCFGLOCKR) 1638
32.9.4 FMC resource x CID register (FMC_CIDCFGRx) 1639
32.9.5 FMC resource x semaphore control register (FMC_SEMCRx) 1640
32.10 NOR flash/PSRAM controller 1641
32.10.1 External memory interface signals 1643
32.10.2 Supported memories and transactions 1645
32.10.3 General timing rules 1647
32.10.4 NOR flash/PSRAM controller asynchronous transactions 1647
32.10.5 Synchronous transactions 1666
32.10.6 NOR/PSRAM controller registers 1671
32.11 NAND flash controller 1684
32.11.1 External memory interface signals 1685
32.11.2 NAND flash supported memories and transactions 1686
32.11.3 Timing diagrams for NAND flash memory 1686
32.11.4 NAND flash operations 1688
32.11.5 NAND flash prewait function 1689
32.11.6 NAND ECC controller 1689
32.11.7 FMC command sequencer 1692
32.11.8 NAND flash controller interrupt 1699
32.11.9 NAND flash controller registers 1700
32.12 FMC register map 1724

33 Octo-SPI interface (OCTOSPI) 1729
33.1 OCTOSPI introduction 1729
33.2 OCTOSPI main features 1729
33.3 OCTOSPI implementation 1730
33.4 OCTOSPI functional description 1731
33.4.1 OCTOSPI block diagram 1731
33.4.2 OCTOSPI pins and internal signals 1732
33.4.3 OCTOSPI interface to memory modes 1733
33.4.4 OCTOSPI regular-command protocol 1733
33.4.5 OCTOSPI regular-command protocol signal interface 1737
33.4.6 HyperBus protocol 1740
33.4.7 Specific features 1744
33.4.8 OCTOSPI operating mode introduction 1745
33.4.9 OCTOSPI indirect mode 1745

33.4.10 OCTOSPI automatic status-polling mode 1747
33.4.11 OCTOSPI memory-mapped mode 1748
33.4.12 OCTOSPI configuration introduction 1749
33.4.13 OCTOSPI system configuration 1749
33.4.14 OCTOSPI device configuration 1750
33.4.15 OCTOSPI regular-command mode configuration 1752
33.4.16 OCTOSPI HyperBus protocol configuration 1755
33.4.17 OCTOSPI error management 1755
33.4.18 OCTOSPI BUSY and ABORT 1756
33.4.19 OCTOSPI reconfiguration or deactivation 1756
33.4.20 NCS behavior 1757
33.4.21 Software control of two external memories 1758
33.5 Address alignment and data number 1759
33.6 OCTOSPI interrupts 1760
33.7 OCTOSPI registers 1760
33.7.1 OCTOSPI control register (OCTOSPI_CR) 1760
33.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1) 1763
33.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2) 1765
33.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3) 1766
33.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4) 1767
33.7.6 OCTOSPI status register (OCTOSPI_SR) 1767
33.7.7 OCTOSPI flag clear register (OCTOSPI_FCR) 1768
33.7.8 OCTOSPI data length register (OCTOSPI_DLR) 1769
33.7.9 OCTOSPI address register (OCTOSPI_AR) 1769
33.7.10 OCTOSPI data register (OCTOSPI_DR) 1770
33.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR) 1770
33.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR) 1771
33.7.13 OCTOSPI polling interval register (OCTOSPI_PIR) 1771
33.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR) . 1772
33.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR) 1774
33.7.16 OCTOSPI instruction register (OCTOSPI_IR) 1774
33.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR) 1775
33.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR) 1775
33.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR) 1776
33.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR) . 1778
33.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR) 1778

33.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR) 1779
33.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR) 1779
33.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR) . . . 1781
33.7.25 OCTOSPI write instruction register (OCTOSPI_WIR) 1782
33.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR) 1782
33.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR) 1783
33.7.28 OCTOSPI hardware configuration register 2 (OCTOSPI_HWCFGR2) 1783
33.7.29 OCTOSPI hardware configuration register (OCTOSPI_HWCFGR) . 1784
33.7.30 OCTOSPI version register (OCTOSPI_VERR) 1785
33.7.31 OCTOSPI identification register (OCTOSPI_IDR) 1785
33.7.32 OCTOSPI hardware magic identification register (OCTOSPI_MIDR) 1786
33.7.33 OCTOSPI register map 1786

34 Secure digital input/output MultiMediaCard interface (SDMMC) . . 1790

34.1 SDMMC main features 1790
34.2 SDMMC implementation 1790
34.3 SDMMC bus topology 1791
34.4 SDMMC operation modes 1793
34.5 SDMMC functional description 1794
34.5.1 SDMMC block diagram 1794
34.5.2 SDMMC pins and internal signals 1794
34.5.3 General description 1795
34.5.4 SDMMC adapter 1797
34.5.5 SDMMC AHB slave interface 1819
34.5.6 SDMMC AHB master interface 1820
34.5.7 AHB and SDMMC_CK clock relation 1823
34.6 Card functional description 1824
34.6.1 SD I/O mode 1824
34.6.2 CMD12 send timing 1832
34.6.3 Sleep (CMD5) 1836
34.6.4 Interrupt mode (Wait-IRQ) 1837
34.6.5 Boot operation 1838
34.6.6 Response R1b handling 1841
34.6.7 Reset and card cycle power 1842

34.7 Hardware flow control 1843
34.7.1 Hardware flow control during data transfer 1843
34.7.2 Block gap hardware flow control 1844
34.8 Ultra-high-speed phase I (UHS-I) voltage switch 1845
34.9 SDMMC interrupts 1848
34.10 SDMMC registers 1850
34.10.1 SDMMC power control register (SDMMC_POWER) 1850
34.10.2 SDMMC clock control register (SDMMC_CLKCR) 1851
34.10.3 SDMMC argument register (SDMMC_ARGR) 1853
34.10.4 SDMMC command register (SDMMC_CMDR) 1853
34.10.5 SDMMC command response register (SDMMC_RESPCMDR) 1855
34.10.6 SDMMC response x register (SDMMC_RESPxR) 1856
34.10.7 SDMMC data timer register (SDMMC_DTIMER) 1856
34.10.8 SDMMC data length register (SDMMC_DLENR) 1857
34.10.9 SDMMC data control register (SDMMC_DCTRL) 1858
34.10.10 SDMMC data counter register (SDMMC_DCNTR) 1859
34.10.11 SDMMC status register (SDMMC_STAR) 1860
34.10.12 SDMMC interrupt clear register (SDMMC_ICR) 1863
34.10.13 SDMMC mask register (SDMMC_MASKR) 1865
34.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 1868
34.10.15 SDMMC data FIFO threshold register (SDMMC_FIFOTHRR) 1868
34.10.16 SDMMC DMA control register (SDMMC_IDMACTRLR) 1869
34.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) 1869
34.10.18 SDMMC IDMA buffer base address register (SDMMC_IDMABASER) 1870
34.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR) . . . 1870
34.10.20 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR) 1871
34.10.21 SDMMC data FIFO registers x (SDMMC_FIFORx) 1872
34.10.22 SDMMC hardware configuration register 1 (SDMMC_HWCFGR1) . 1872
34.10.23 SDMMC version register (SDMMC_VERR) 1873
34.10.24 SDMMC identification register (SDMMC_IPIDR) 1873
34.10.25 SDMMC size identification register (SDMMC_SIDR) 1873
34.10.26 SDMMC register map 1874

35 Delay block (DLYB) 1877

35.1 DLYB introduction 1877

35.2 DLYB main features . . . 1877
35.3 DLYB implementation . . . 1877
35.4 DLYB functional description . . . 1877
35.4.1 DLYB diagram . . . 1877
35.4.2 DLYB internal signals . . . 1879
35.4.3 General description . . . 1881
35.4.4 Lock mode procedure . . . 1882
35.4.5 Bypass mode procedure . . . 1882

36 Analog-to-digital converters (ADC) . . . 1883
36.1 ADC introduction . . . 1883
36.2 ADC main features . . . 1883
36.3 ADC implementation . . . 1885
36.4 ADC functional description . . . 1886
36.4.1 ADC block diagram . . . 1886
36.4.2 ADC pins and internal signals . . . 1887
36.4.3 ADC clocks . . . 1892
36.4.4 ADC connectivity . . . 1894
36.4.5 Slave AHB interface . . . 1896
36.4.6 ADC Deep-power-down mode (DEEPPWD) . . . 1896
36.4.7 Single-ended and differential input channels . . . 1896
36.4.8 Calibration (ADCAL, CALADDOS, ADC_CALFACT) . . . 1897
36.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 1899
36.4.10 Constraints when writing the ADC control bits . . . 1900
36.4.11 Channel selection (ADC_SQRy, ADC_JSQR) . . . 1901
36.4.12 Channel preselection register (ADC_PCSEL) . . . 1901
36.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 1902
36.4.14 Single conversion mode . . . 1903
36.4.15 Continuous conversion mode (CONT = 1) . . . 1904
36.4.16 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . 1905
36.4.17 Starting conversions (ADSTART, JADSTART) . . . 1906
36.4.18 Timing . . . 1907
36.4.19 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 1908
36.4.20 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . 1910
36.4.21 Injected channel management . . . 1911
36.4.22 Programmable resolution (RES) - fast conversion mode . . . 1912

36.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) . . . 1912
36.4.24 End of conversion sequence (EOS, JEOS) . . . 1912
36.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers) . . . 1913
36.4.26 Data management . . . 1915
36.4.27 Managing conversions using the MDF . . . 1923
36.4.28 Dynamic low-power features . . . 1923
36.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) . . . 1927
36.4.30 Oversampler . . . 1932
36.4.31 VBAT supply monitoring . . . 1939
36.4.32 Monitoring the internal voltage reference . . . 1940
36.4.33 Monitoring the supply voltage . . . 1942
36.5 ADC interrupts . . . 1942
36.6 ADC registers (for each ADC) . . . 1943
36.6.1 ADC interrupt and status register (ADC_ISR) . . . 1943
36.6.2 ADC interrupt enable register (ADC_IER) . . . 1945
36.6.3 ADC control register (ADC_CR) . . . 1947
36.6.4 ADC configuration register (ADC_CFGR1) . . . 1950
36.6.5 ADC configuration register 2 (ADC_CFGR2) . . . 1953
36.6.6 ADC sample time register 1 (ADC_SMPR1) . . . 1956
36.6.7 ADC sample time register 2 (ADC_SMPR2) . . . 1957
36.6.8 ADC channel preselection register (ADC_PCSEL) . . . 1958
36.6.9 ADC regular sequence register 1 (ADC_SQR1) . . . 1958
36.6.10 ADC regular sequence register 2 (ADC_SQR2) . . . 1959
36.6.11 ADC regular sequence register 3 (ADC_SQR3) . . . 1960
36.6.12 ADC regular sequence register 4 (ADC_SQR4) . . . 1961
36.6.13 ADC regular data register (ADC_DR) . . . 1962
36.6.14 ADC injected sequence register (ADC_JSQR) . . . 1962
36.6.15 ADC offset y configuration register (ADC_OFCFGRy) . . . 1964
36.6.16 ADC offset y register (ADC_OFRy) . . . 1965
36.6.17 ADC gain compensation register (ADC_GCOMP) . . . 1966
36.6.18 ADC injected channel y data register (ADC_JDRy) . . . 1967
36.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 1967
36.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 1968
36.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) 1968

36.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) . . . 1969
36.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) 1969
36.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) . . . 1970
36.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) 1970
36.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) . . . 1970
36.6.27 ADC differential mode selection register (ADC_DIFSEL) . . . 1971
36.6.28 ADC calibration factors (ADC_CALFACT) . . . 1971
36.6.29 ADC option register (ADC_OR) . . . 1973
36.7 ADC common registers . . . 1974
36.7.1 ADC common status register (ADCC_CSR) . . . 1974
36.7.2 ADC common control register (ADCC_CCR) . . . 1975
36.7.3 ADC hardware configuration register (ADCC_HWCFGR1) . . . 1976
36.7.4 ADC version register (ADCC_VERR) . . . 1976
36.7.5 ADC identification register (ADCC_IPIDR) . . . 1977
36.7.6 ADC size identification register (ADCC_SIDR) . . . 1977
36.8 ADC register map . . . 1978

37 Digital temperature sensor (DTS) . . . 1982
37.1 Introduction . . . 1982
37.2 DTS main features . . . 1982
37.3 DTS functional description . . . 1982
37.3.1 DTS block diagram . . . 1982
37.3.2 DTS pins and internal signals . . . 1983
37.3.3 DTS reset and clocks . . . 1983
37.3.4 DTS serial data adapter (SDA) . . . 1985
37.3.5 Alarms . . . 1991
37.3.6 Main APB programming routines . . . 1992
37.3.7 Temperature sensor configuration . . . 1995
37.4 DTS interrupts . . . 1997
37.5 DTS registers . . . 1998
37.5.1 DTS PVT common registers . . . 1998
37.5.2 PVT IRQ registers . . . 2000
37.5.3 TS common registers . . . 2004

37.5.4 TS individual registers 2011
37.5.5 DTS register map 2018

38 Voltage reference buffer (VREFBUF) 2023
38.1 VREFBUF introduction 2023
38.2 VREFBUF implementation 2023
38.3 VREFBUF functional description 2023
38.4 VREFBUF trimming 2024
38.5 VREFBUF power sequence 2025
38.6 VREFBUF registers 2026
38.6.1 VREFBUF control and status register (VREFBUF_CSR) 2026
38.6.2 VREFBUF calibration control register (VREFBUF_CCR) 2027
38.6.3 VREFBUF register map 2027

39 Multi-function digital filter (MDF) 2028
39.1 MDF introduction 2028
39.2 MDF main features 2029
39.3 MDF implementation 2029
39.4 MDF functional description 2031
39.4.1 MDF block diagram 2031
39.4.2 MDF pins and internal signals 2032
39.4.3 Serial input interfaces (SITF) 2033
39.4.4 ADC slave interface (ADCITF) 2039
39.4.5 Clock generator (CKGEN) 2040
39.4.6 Bitstream matrix (BSMX) 2042
39.4.7 Short-circuit detectors (SCD) 2043
39.4.8 Digital filter processing (DFLT) 2045
39.4.9 Out-of-limit detector (OLD) 2055
39.4.10 Digital filter acquisition modes 2058
39.4.11 Start-up sequence examples 2068
39.4.12 Break interface 2069
39.4.13 Data transfer to memory 2070
39.4.14 Autonomous mode 2075
39.4.15 Register protection 2076
39.5 MDF low-power modes 2077
39.6 MDF interrupts 2077

39.7 MDF application informations 2079
39.7.1 MDF configuration examples for audio capture 2079
39.7.2 Programming examples 2080
39.7.3 Connection examples 2082
39.7.4 Global frequency response 2083
39.7.5 Total MDF gain 2084
39.8 MDF registers 2088
39.8.1 MDF global control register (MDF_GCR) 2089
39.8.2 MDF clock generator control register (MDF_CKGCR) 2089
39.8.3 MDF serial interface control register x (MDF_SITFxCR) 2092
39.8.4 MDF bitstream matrix control register x (MDF_BSMXxCR) 2093
39.8.5 MDF digital filter control register x (MDF_DFLTxCR) 2094
39.8.6 MDF digital filter configuration register x (MDF_DFLTxCICR) 2096
39.8.7 MDF reshape filter configuration register x (MDF_DFLTxRSFR) . . . 2097
39.8.8 MDF integrator configuration register x (MDF_DFLTxINTR) 2098
39.8.9 MDF out-of limit detector control register x (MDF_OLDxCR) 2099
39.8.10 MDF OLDx low threshold register x (MDF_OLDxTHLR) 2101
39.8.11 MDF OLDx high threshold register x (MDF_OLDxTHHR) 2101
39.8.12 MDF delay control register x (MDF_DLYxCR) 2102
39.8.13 MDF short circuit detector control register x (MDF_SCDxCR) 2102
39.8.14 MDF DFLT0 interrupt enable register 0 (MDF_DFLT0IER) 2103
39.8.15 MDF DFLTx interrupt enable register x (MDF_DFLTxIER) 2105
39.8.16 MDF DFLT0 interrupt status register 0 (MDF_DFLT0ISR) 2106
39.8.17 MDF DFLTx interrupt status register x (MDF_DFLTxISR) 2108
39.8.18 MDF offset error compensation control register x (MDF_OECxCR) . 2109
39.8.19 MDF snapshot data register x (MDF_SNPSxDR) 2110
39.8.20 MDF digital filter data register x (MDF_DFLTxDR) 2110
39.8.21 MDF hardware configuration register (MDF_HWCFGR) 2111
39.8.22 MDF version register (MDF_VERR) 2112
39.8.23 MDF identification register (MDF_IPIDR) 2112
39.8.24 MDF size identification register (MDF_SIDR) 2112
39.8.25 MDF register map 2113

40 Camera subsystem 2116
40.1 Introduction 2116
40.2 Camera path main features 2116
40.2.1 DCMI path main features 2116

40.2.2 DCMIPP path main features 2117
40.3 Camera subsystem implementation 2118
40.3.1 Hardware settings 2118
40.4 Camera subsystem functional description 2119
40.4.1 Camera subsystem block diagram 2119
40.4.2 Camera subsystem pins and external signal interface 2119
40.4.3 Camera subsystem reset and clocks 2120
40.4.4 Camera subsystem security 2120
40.5 Camera subsystem programmable parameters 2122
40.5.1 Low-resolution parallel camera and DCMI 2122
40.5.2 High-resolution parallel camera and DCMIPP 2123
40.5.3 High-resolution CSI2 camera and DCMIPP 2123
40.5.4 Sensors over target resolution 2124
40.5.5 Sensors over target pixel rate 2124
40.5.6 Double sensors 2125
40.6 Camera subsystem interrupts 2125
40.7 Camera subsystem registers 2125

41 Digital camera interface (DCMI) 2126
41.1 DCMI introduction 2126
41.2 DCMI main features 2126
41.3 DCMI functional description 2126
41.3.1 DCMI block diagram 2127
41.3.2 DCMI pins and internal signals 2127
41.3.3 DCMI clocks 2128
41.3.4 DCMI DMA interface 2128
41.3.5 DCMI physical interface 2128
41.3.6 DCMI synchronization 2130
41.3.7 DCMI capture modes 2132
41.3.8 DCMI crop feature 2133
41.3.9 DCMI JPEG format 2134
41.3.10 DCMI FIFO 2134
41.3.11 DCMI data format description 2135
41.4 DCMI interrupts 2137
41.5 DCMI registers 2137
41.5.1 DCMI control register (DCMI_CR) 2137

41.5.2 DCMI status register (DCMI_SR) 2140
41.5.3 DCMI raw interrupt status register (DCMI_RIS) 2140
41.5.4 DCMI interrupt enable register (DCMI_IER) 2141
41.5.5 DCMI masked interrupt status register (DCMI_MIS) 2142
41.5.6 DCMI interrupt clear register (DCMI_ICR) 2143
41.5.7 DCMI embedded synchronization code register (DCMI_ESCR) 2144
41.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . 2144
41.5.9 DCMI crop window start (DCMI_CWSTRT) 2145
41.5.10 DCMI crop window size (DCMI_CWSIZE) 2146
41.5.11 DCMI data register (DCMI_DR) 2146
41.5.12 DCMI hardware configuration register (DCMI_HWCFGR) 2147
41.5.13 DCMI version register (DCMI_VERR) 2147
41.5.14 DCMI identification register (DCMI_IPIDR) 2147
41.5.15 DCMI size identification register (DCMI_SIDR) 2148
41.5.16 DCMI register map 2148

42 Digital camera interface pixel pipeline (DCMIPP) 2150

42.1 DCMIPP introduction 2150
42.2 DCMIPP main features 2153
42.3 DCMIPP functional description 2155
42.3.1 DCMIPP block diagram 2155
42.3.2 DCMIPP pads and internal signals 2156
42.3.3 DCMIPP reset and clocks 2156
42.3.4 DCMIPP maximum resolution 2159
42.3.5 DCMIPP minimum requirements for frame structure 2160
42.3.6 Description of DCMIPP pixel format support 2160
42.4 DCMIPP input and flow control 2161
42.4.1 DCMIPP common configuration 2162
42.4.2 Parallel input interface 2162
42.4.3 Interface from CSI-2 host 2166
42.4.4 Test pattern generator (TPG) 2166
42.4.5 Input selection 2168
42.4.6 Flow selection 2168
42.4.7 Frame counter 2170
42.4.8 Frame control 2170
42.4.9 Pipe deactivation 2176
42.5 Pipe0 (dump pipe) 2177

42.5.1 Overview . . . 2177
42.5.2 Decimation . . . 2178
42.5.3 Crop/statistics selection/suppression . . . 2178
42.5.4 Header insertion . . . 2179
42.5.5 Dump counter . . . 2180
42.6 Pipe1 (ISP part) . . . 2181
42.6.1 Overview . . . 2181
42.6.2 Byte-to-pixel conversion . . . 2182
42.6.3 Statistics removal . . . 2183
42.6.4 Bad pixel removal . . . 2183
42.6.5 Input decimation . . . 2185
42.6.6 Black level calibration . . . 2186
42.6.7 Exposure compensation and white-balance calibration . . . 2186
42.6.8 Demosaicing . . . 2188
42.6.9 Color conversion . . . 2190
42.6.10 Contrast enhancement . . . 2192
42.6.11 Statistics extraction . . . 2194
42.6.12 Histogram . . . 2197
42.7 Pipe1 (post-processing part) . . . 2204
42.7.1 Overview . . . 2204
42.7.2 Pixel 2D cropping . . . 2204
42.7.3 Decimation pre-downsize . . . 2205
42.7.4 Downsize . . . 2206
42.7.5 Gamma conversion . . . 2208
42.7.6 YUV conversion . . . 2208
42.7.7 Chroma down-sampling . . . 2208
42.7.8 Pixel packing . . . 2209
42.7.9 Overrun detection . . . 2210
42.8 Pipe2 (post-processing) . . . 2211
42.8.1 Overview . . . 2211
42.8.2 Pipe1 and Pipe2 sharing image processing functions . . . 2211
42.9 Application use cases . . . 2212
42.9.1 Parallel interface camera sensor module . . . 2212
42.9.2 CSI2 camera sensor module . . . 2213
42.9.3 Force data type format from CSI-2 data flow (pixel pipes only) . . . 2219
42.10 Pixel format description . . . 2222

42.10.1 Parallel interface formats . . . 2222
42.10.2 Pixel pipe formats . . . 2224
42.10.3 Dump pipe formats . . . 2225
42.10.4 AXI IP-Plug . . . 2227
42.11 Shadow registers . . . 2233
42.12 DCMIPP low power modes . . . 2235
42.13 DCMIPP interrupts . . . 2236
42.13.1 Free-running DCMIPP . . . 2236
42.13.2 Interrupts . . . 2236
42.13.3 Event pins . . . 2239
42.14 DCMIPP registers . . . 2241
42.14.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1) . . . 2241
42.14.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2) . . . 2242
42.14.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3) . . . 2242
42.14.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8) . . . 2243
42.14.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1) . . . 2243
42.14.6 DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2) . . . 2244
42.14.7 DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3) . . . 2244
42.14.8 DCMIPP parallel interface control register (DCMIPP_PRCR) . . . 2245
42.14.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESCR) . . . 2246
42.14.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR) . . . 2247
42.14.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER) . . . 2248
42.14.12 DCMIPP parallel interface status register (DCMIPP_PRSR) . . . 2248
42.14.13 DCMIPP parallel interface interrupt clear register (DCMIPP_PRFCR) . . . 2249
42.14.14 DCMIPP common IP HW configuration register (DCMIPP_CMHWCFGR) . . . 2250
42.14.15 DCMIPP common configuration register (DCMIPP_CMCR) . . . 2250
42.14.16 DCMIPP common frame counter register (DCMIPP_CMFRCR) . . . 2251
42.14.17 DCMIPP common test pattern generator configuration register 1 (DCMIPP_CMTPGCR1) . . . 2251
42.14.18 DCMIPP common test pattern generator configuration register 2 (DCMIPP_CMTPGCR2) . . . 2252
42.14.19 DCMIPP common interrupt enable register (DCMIPP_CMIER) . . . 2253
42.14.20 DCMIPP common status register 1 (DCMIPP_CMSR1) . . . 2255
42.14.21 DCMIPP common status register 2 (DCMIPP_CMSR2) . . . 2256

42.14.22 DCMIPP common interrupt clear register (DCMIPP_CMFCR) 2259
42.14.23 DCMIPP Pipe0 HW configuration register (DCMIPP_P0HWCFGR) . 2260
42.14.24 DCMIPP Pipe0 flow selection configuration register (DCMIPP_P0FSCR) 2261
42.14.25 DCMIPP Pipe0 flow control configuration register (DCMIPP_P0FCTCR) 2262
42.14.26 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR) . . . 2262
42.14.27 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR) . . . 2263
42.14.28 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR) 2264
42.14.29 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR) 2264
42.14.30 DCMIPP Pipe0 pixel packer configuration register (DCMIPP_P0PPCR) 2265
42.14.31 DCMIPP Pipe0 pixel packer Memory0 address register 1 (DCMIPP_P0PPM0AR1) 2266
42.14.32 DCMIPP Pipe0 status Memory0 address register (DCMIPP_P0STM0AR) 2266
42.14.33 DCMIPP Pipe0 interrupt enable register (DCMIPP_P0IER) 2267
42.14.34 DCMIPP Pipe0 status register (DCMIPP_P0SR) 2267
42.14.35 DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR) 2269
42.14.36 DCMIPP Pipe0 current flow selection configuration register (DCMIPP_P0CFSCR) 2269
42.14.37 DCMIPP Pipe0 current flow control configuration register (DCMIPP_P0CFCTCR) 2270
42.14.38 DCMIPP Pipe0 current statistic/crop start register (DCMIPP_P0CSCSTR) 2271
42.14.39 DCMIPP Pipe0 current statistic/crop size register (DCMIPP_P0CSCSZR) 2271
42.14.40 DCMIPP Pipe0 current pixel packer configuration register (DCMIPP_P0CPPCR) 2272
42.14.41 DCMIPP Pipe0 current pixel packer Memory0 address register 1 (DCMIPP_P0CPPM0AR1) 2274
42.14.42 DCMIPP Pipex HW configuration register (DCMIPP_PxHWCFGR) . 2274
42.14.43 DCMIPP Pipe1 flow selection configuration register (DCMIPP_P1FSCR) 2275
42.14.44 DCMIPP Pipe1 stat removal configuration register (DCMIPP_P1SRCR) 2276
42.14.45 DCMIPP Pipe1 bad pixel removal control register (DCMIPP_P1BPRCR) 2277
42.14.46 DCMIPP Pipe1 bad pixel removal status register (DCMIPP_P1BPRSR) 2277
42.14.47 DCMIPP Pipe1 decimation register (DCMIPP_P1DECR) 2278

42.14.48 DCMIPP Pipe1 black level calibration control register (DCMIPP_P1BLCCR) 2278
42.14.49 DCMIPP Pipe1 exposure control register 1 (DCMIPP_P1EXCR1) . . 2279
42.14.50 DCMIPP Pipe1 exposure control register 2 (DCMIPP_P1EXCR2) . . 2279
42.14.51 DCMIPP Pipe1 statistics1 control register (DCMIPP_P1ST1CR) . . . 2280
42.14.52 DCMIPP Pipe1 statistics 2 control register (DCMIPP_P1ST2CR) . . . 2281
42.14.53 DCMIPP Pipe1 statistics 3 control register (DCMIPP_P1ST3CR) . . . 2282
42.14.54 DCMIPP Pipe1 statistics window start register (DCMIPP_P1STSTR) 2283
42.14.55 DCMIPP Pipe1 statistics window size register (DCMIPP_P1STSZR) 2283
42.14.56 DCMIPP Pipe1 statistics 1 status register (DCMIPP_P1ST1SR) . . . 2284
42.14.57 DCMIPP Pipe1 statistics 2 status register (DCMIPP_P1ST2SR) . . . 2285
42.14.58 DCMIPP Pipe1 statistics 3 status register (DCMIPP_P1ST3SR) . . . 2285
42.14.59 DCMIPP Pipe1 demosaicing configuration register (DCMIPP_P1DMCR) 2285
42.14.60 DCMIPP Pipe1 ColorConv configuration register (DCMIPP_P1CCCR) 2286
42.14.61 DCMIPP Pipe1 ColorConv red coefficient register 1 (DCMIPP_P1CCRR1) 2287
42.14.62 DCMIPP Pipe1 ColorConv red coefficient register 2 (DCMIPP_P1CCRR2) 2287
42.14.63 DCMIPP Pipe1 ColorConv green coefficient register 1 (DCMIPP_P1CCGR1) 2288
42.14.64 DCMIPP Pipe1 ColorConv green coefficient register 2 (DCMIPP_P1CCGR2) 2288
42.14.65 DCMIPP Pipex ColorConv blue coefficient register 1 (DCMIPP_P1CCBR1) 2289
42.14.66 DCMIPP Pipe1 ColorConv blue coefficient register 2 (DCMIPP_P1CCBR2) 2289
42.14.67 DCMIPP Pipe1 contrast control register 1 (DCMIPP_P1CTCR1) . . . 2290
42.14.68 DCMIPP Pipe1 contrast control register 2 (DCMIPP_P1CTCR2) . . . 2290
42.14.69 DCMIPP Pipe1 contrast control register 3 (DCMIPP_P1CTCR3) . . . 2291
42.14.70 DCMIPP Pipe1 histogram configuration register (DCMIPP_P1HSCR) . . . 2291
42.14.71 DCMIPP Pipe1 histogram window start register (DCMIPP_P1HSSTR) 2293
42.14.72 DCMIPP Pipe1 histogram window size register (DCMIPP_P1HSSZR) 2294
42.14.73 DCMIPP Pipe1 histogram memory address register 1 (DCMIPP_P1HSMAR1) 2294

42.14.74 DCMIPP Pipe1 histogram memory status address register 3 (DCMIPP_P1HSSMAR) . . . 2295
42.14.75 DCMIPP Pipex flow control configuration register (DCMIPP_PxFCTCR) . . . 2295
42.14.76 DCMIPP Pipex crop window start register (DCMIPP_PxCRSTR) . . . 2296
42.14.77 DCMIPP Pipex crop window size register (DCMIPP_PxCRSZR) . . . 2296
42.14.78 DCMIPP Pipex decimation register (DCMIPP_PxDCCR) . . . 2297
42.14.79 DCMIPP Pipex downsize configuration register (DCMIPP_PxDSCR) . . . 2297
42.14.80 DCMIPP Pipex downsize ratio register (DCMIPP_PxDSRTIOR) . . . 2298
42.14.81 DCMIPP Pipex downsize destination size register (DCMIPP_PxDSSZR) . . . 2298
42.14.82 DCMIPP Pipex gamma configuration register (DCMIPP_PxGMCR) . 2299
42.14.83 DCMIPP Pipe1 YUVConv configuration register (DCMIPP_P1YUVCR) . . . 2299
42.14.84 DCMIPP Pipe1 YUVConv red coefficient register 1 (DCMIPP_P1YUVRR1) . . . 2300
42.14.85 DCMIPP Pipe1 YUVConv red coefficient register 2 (DCMIPP_P1YUVRR2) . . . 2300
42.14.86 DCMIPP Pipe1 YUVConv green coefficient register 1 (DCMIPP_P1YUVGR1) . . . 2300
42.14.87 DCMIPP Pipe1 YUVConv green coefficient register 2 (DCMIPP_P1YUVGR2) . . . 2301
42.14.88 DCMIPP Pipe1 YUVConv blue coefficient register 1 (DCMIPP_P1YUVBR1) . . . 2301
42.14.89 DCMIPP Pipe1 YUV blue coefficient register 2 (DCMIPP_P1YUVBR2) . . . 2302
42.14.90 DCMIPP Pipe1 pixel packer configuration register (DCMIPP_P1PPCR) . . . 2302
42.14.91 DCMIPP Pipe1 pixel packer Memory0 address register 1 (DCMIPP_P1PPM0AR1) . . . 2303
42.14.92 DCMIPP Pipex pixel packer Memory0 pitch register (DCMIPP_PxPPM0PR) . . . 2303
42.14.93 DCMIPP Pipex status Memory0 address register (DCMIPP_PxSTM0AR) . . . 2304
42.14.94 DCMIPP Pipex pixel packer Memory1 address register 1 (DCMIPP_PxPPM1AR1) . . . 2304
42.14.95 DCMIPP Pipex pixel packer Memory1 pitch register (DCMIPP_PxPPM1PR) . . . 2305
42.14.96 DCMIPP Pipex status Memory1 address register (DCMIPP_PxSTM1AR) . . . 2305
42.14.97 DCMIPP Pipex pixel packer memory2 address register 1 (DCMIPP_PxPPM2AR1) . . . 2306

42.14.98 DCMIPP Pipex status Memory2 address register (DCMIPP_PxSTM2AR) 2306
42.14.99 DCMIPP Pipe1 interrupt enable register (DCMIPP_P1IER) 2306
42.14.100 DCMIPP Pipe1 status register (DCMIPP_P1SR) 2307
42.14.101 DCMIPP Pipe1 interrupt clear register (DCMIPP_P1FCR) 2309
42.14.102 DCMIPP Pipe1 current flow selection configuration register (DCMIPP_P1CFSCR) 2309
42.14.103 DCMIPP Pipe1 current bad pixel removal register (DCMIPP_P1CBPRCR) 2310
42.14.104 DCMIPP Pipe1 current black level calibration control register (DCMIPP_P1CBLCCR) 2311
42.14.105 DCMIPP Pipe1 current exposure control register 1 (DCMIPP_P1CEXCR1) 2311
42.14.106 DCMIPP Pipe1 current exposure control register 2 (DCMIPP_P1CEXCR2) 2312
42.14.107 DCMIPP Pipe1 current statistics 1 control register (DCMIPP_P1CST1CR) 2313
42.14.108 DCMIPP Pipe1 current statistics 2 control register (DCMIPP_P1CST2CR) 2314
42.14.109 DCMIPP Pipe1 current statistics 3 control register (DCMIPP_P1CST3CR) 2315
42.14.110 DCMIPP Pipe1 current statistics window start register (DCMIPP_P1CSTSTR) 2316
42.14.111 DCMIPP Pipe1 current statistics window size register (DCMIPP_P1CSTSZR) 2316
42.14.112 DCMIPP Pipe1 current ColorConv configuration register (DCMIPP_P1CCCCR) 2317
42.14.113 DCMIPP Pipe1 current ColorConv red coefficient register 1 (DCMIPP_P1CCCRR1) 2317
42.14.114 DCMIPP Pipe1 current ColorConv red coefficient register 2 (DCMIPP_P1CCCRR2) 2318
42.14.115 DCMIPP Pipe1 current ColorConv green coefficient register 1 (DCMIPP_P1CCCGR1) 2318
42.14.116 DCMIPP Pipe1 current ColorConv green coefficient register 2 (DCMIPP_P1CCCGR2) 2319
42.14.117 DCMIPP Pipex current ColorConv blue coefficient register 1 (DCMIPP_P1CCCBR1) 2319
42.14.118 DCMIPP Pipe1 current ColorConv blue coefficient register 2 (DCMIPP_P1CCCBR2) 2320
42.14.119 DCMIPP Pipe1 current contrast control register 1 (DCMIPP_P1CCTCR1) 2320
42.14.120 DCMIPP Pipe1 current contrast control register 2 (DCMIPP_P1CCTCR2) 2321

42.14.121DCMIPP Pipe1 current contrast control register 3 (DCMIPP_P1CCTCR3) 2321
42.14.122DCMIPP Pipe1 current histogram configuration register (DCMIPP_P1CHSCR) 2322
42.14.123DCMIPP Pipe1 current histogram window start register (DCMIPP_P1CHSSTR) 2324
42.14.124DCMIPP Pipe1 current histogram window size register (DCMIPP_P1CHSSZR) 2324
42.14.125DCMIPP Pipe1 current histogram memory address register 1 (DCMIPP_P1CHSMAR1) 2325
42.14.126DCMIPP Pipex current flow control configuration register (DCMIPP_PxCFCTCR) 2325
42.14.127DCMIPP Pipex current crop window start register (DCMIPP_PxCCRSTR) 2326
42.14.128DCMIPP Pipex current crop window size register (DCMIPP_PxCCRSZR) 2326
42.14.129DCMIPP Pipex current decimation register (DCMIPP_PxCDCCR) . 2327
42.14.130DCMIPP Pipex current downsize configuration register (DCMIPP_PxCDSCR) 2327
42.14.131DCMIPP Pipex current downsize ratio register (DCMIPP_PxCDSRTIOR) 2328
42.14.132DCMIPP Pipex current downsize destination size register (DCMIPP_PxCDSSZR) 2328
42.14.133DCMIPP Pipe1 current pixel packer configuration register (DCMIPP_P1CPPCR) 2329
42.14.134DCMIPP Pipe1 current pixel packer Memory0 address register 1 (DCMIPP_P1CPPM0AR1) 2329
42.14.135DCMIPP Pipex current pixel packer Memory0 pitch register (DCMIPP_PxCPPM0PR) 2330
42.14.136DCMIPP Pipex current pixel packer Memory1 address register 1 (DCMIPP_PxCPPM1AR1) 2330
42.14.137DCMIPP Pipex current pixel packer Memory1 pitch register (DCMIPP_PxCPPM1PR) 2331
42.14.138DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR1) 2331
42.14.139DCMIPP Pipe2 flow selection configuration register (DCMIPP_P2FSCR) 2332
42.14.140DCMIPP Pipe2 pixel packer configuration register (DCMIPP_P2PPCR) 2333
42.14.141DCMIPP Pipe2 pixel packer Memory0 address register 1 (DCMIPP_P2PPM0AR1) 2334
42.14.142DCMIPP Pipe2 interrupt enable register (DCMIPP_P2IER) 2334
42.14.143DCMIPP Pipe2 status register (DCMIPP_P2SR) 2335

42.14.144DCMIPP Pipe2 interrupt clear register (DCMIPP_P2FCR) 2336
42.14.145DCMIPP Pipe2 current flow selection configuration register (DCMIPP_P2CFSCR) 2336
42.14.146DCMIPP Pipe2 current pixel packer configuration register (DCMIPP_P2CPPCR) 2337
42.14.147DCMIPP Pipe2 current pixel packer Memory0 address register 1 (DCMIPP_P2CPPM0AR1) 2338
42.14.148DCMIPP hardware configuration register 3 (DCMIPP_HWCFGR3) . 2339
42.14.149DCMIPP hardware configuration register 2 (DCMIPP_HWCFGR2) . 2339
42.14.150DCMIPP hardware configuration register 1 (DCMIPP_HWCFGR1) . 2340
42.14.151DCMIPP version register (DCMIPP_VERR) 2341
42.14.152DCMIPP identification register (DCMIPP_IPIDR) 2341
42.14.153DCMIPP size identification register (DCMIPP_SIDR) 2342
42.15 DCMIPP register map 2343

43 CSI-2 Host (CSI) 2357
43.1 CSI introduction 2357
43.2 Standard and references 2357
43.3 Glossary 2357
43.4 CSI-2 Host main features 2358
43.5 CSI-2 Host functional description 2358
43.5.1 General description 2358
43.5.2 System level architecture 2360
43.5.3 CSI-2 Host reset and clocks 2361
43.5.4 Lane merger 2362
43.5.5 Clock changer 2363
43.5.6 Low-level protocol (LLP) 2364
43.5.7 CSI-2 Host protocol output 2374
43.6 CSI-2 Host programming guide 2374
43.6.1 CSI-2 PHY setup 2374
43.6.2 CSI-2 Host controller setup and data reception start 2375
43.6.3 Data reception stop 2376
43.7 CSI-2 Host low-power modes 2376
43.8 CSI-2 Host interrupts and errors 2376
43.8.1 Error handling 2378
43.8.2 Interrupt handling 2380
43.9 CSI-2 Host registers 2381

43.9.1 CSI-2 Host control register (CSI_CR) 2381
43.9.2 CSI-2 Host DPHY_RX control register (CSI_PCR) 2383
43.9.3 CSI-2 Host virtual channel x configuration register 1 (CSI_VCxCFGR1) 2384
43.9.4 CSI-2 Host virtual channel x configuration register 2 (CSI_VCxCFGR2) 2386
43.9.5 CSI-2 Host virtual channel x configuration register 3 (CSI_VCxCFGR3) 2387
43.9.6 CSI-2 Host virtual channel x configuration register 4 (CSI_VCxCFGR4) 2388
43.9.7 CSI-2 Host line byte x configuration register (CSI_LBxCFGR) 2389
43.9.8 CSI-2 Host timer x configuration register (CSI_TIMxCFGR) 2390
43.9.9 CSI-2 Host lane merger configuration register (CSI_LMCFGR) 2390
43.9.10 CSI-2 Host program interrupt register (CSI_PRGITR) 2391
43.9.11 CSI-2 Host watchdog register (CSI_WDR) 2395
43.9.12 CSI-2 Host interrupt enable register 0 (CSI_IER0) 2395
43.9.13 CSI-2 Host interrupt enable register 1 (CSI_IER1) 2398
43.9.14 CSI-2 Host status register 0 (CSI_SR0) 2400
43.9.15 CSI-2 Host status register 1 (CSI_SR1) 2402
43.9.16 CSI-2 Host flag clear register 0 (CSI_FCR0) 2404
43.9.17 CSI-2 Host flag clear register 1 (CSI_FCR1) 2405
43.9.18 CSI-2 Host short packet data field register (CSI_SPDFR) 2406
43.9.19 CSI-2 Host error register 1 (CSI_ERR1) 2407
43.9.20 CSI-2 Host error register 2 (CSI_ERR2) 2408
43.9.21 CSI-2 Host hardware configuration register (CSI_HHWCFGR) 2408
43.9.22 CSI-2 Host version register (CSI_HVERR) 2409
43.9.23 CSI-2 Host identification register (CSI_HIPIDR) 2409
43.9.24 CSI-2 Host size identification register (CSI_HSIDR) 2410
43.10 CSI PHY registers 2410
43.10.1 CSI PHY reset control register (CSI_PRCR) 2410
43.10.2 CSI PHY mode control register (CSI_PMCR) 2411
43.10.3 CSI PHY frequency control register (CSI_PFCR) 2412
43.10.4 CSI PHY test control register 0 (CSI_PTCR0) 2412
43.10.5 CSI PHY test control register 1 (CSI_PTCR1) 2413
43.10.6 CSI PHY test status register (CSI_PTSR) 2413
43.10.7 CSI-2 Host hardware configuration register (CSI_HWCFGR) 2413
43.10.8 CSI-2 Host version register (CSI_VERR) 2414
43.10.9 CSI-2 Host identification register (CSI_IPIDR) 2414

43.10.10 CSI-2 Host size identification register (CSI_SIDR) 2415
43.10.11 CSI-2 Host and PHY register map 2415

44 Parallel synchronous slave interface (PSSI) 2419
44.1 PSSI introduction 2419
44.2 PSSI main features 2419
44.3 PSSI functional description 2419
44.3.1 PSSI block diagram 2420
44.3.2 PSSI pins and internal signals 2420
44.3.3 PSSI clock 2421
44.3.4 PSSI data management 2421
44.3.5 PSSI optional control signals 2423
44.4 PSSI interrupts 2426
44.5 PSSI registers 2427
44.5.1 PSSI control register (PSSI_CR) 2427
44.5.2 PSSI status register (PSSI_SR) 2428
44.5.3 PSSI raw interrupt status register (PSSI_RIS) 2429
44.5.4 PSSI interrupt enable register (PSSI_IER) 2430
44.5.5 PSSI masked interrupt status register (PSSI_MIS) 2430
44.5.6 PSSI interrupt clear register (PSSI_ICR) 2431
44.5.7 PSSI data register (PSSI_DR) 2431
44.5.8 PSSI hardware configuration register (PSSI_HWCFGR) 2432
44.5.9 PSSI version register (PSSI_VERR) 2432
44.5.10 PSSI ID register (PSSI_IPIDR) 2433
44.5.11 PSSI size identification register (PSSI_SIDR) 2433
44.5.12 PSSI register map 2433

45 Display subsystem 2435
45.1 Introduction 2435
45.2 Display subsystem main features 2435
45.3 Display subsystem implementation 2435
45.4 Display subsystem functional description 2435
45.4.1 Display subsystem block diagram 2435
45.4.2 Display subsystem pins and external signal interface 2436
45.4.3 Display subsystem clocks 2436
45.4.4 Display subsystem security 2436

45.5 Display subsystem programmable parameters 2440
45.5.1 Parallel-interface display 2441
45.6 Display subsystem interrupts 2441
45.7 Display subsystem registers 2442

46 LCD-TFT display controller (LTDC) 2443
46.1 LTDC introduction 2443
46.2 LTDC main features 2443
46.3 LTDC implementation 2443
46.4 LTDC functional description 2444
46.4.1 LTDC block diagram 2444
46.4.2 LTDC pins and internal signals 2445
46.4.3 LTDC reset and clocks 2445
46.5 LTDC configuration parameters 2448
46.5.1 AXI master 2448
46.5.2 Input layer definition and cropping 2448
46.5.3 Input pixel format 2449
46.5.4 YUV planar 2452
46.5.5 YUV-to-RGB color conversion 2452
46.5.6 Horizontal or vertical mirroring 2454
46.5.7 Default layer color 2455
46.5.8 Color look-up table (CLUT) 2455
46.5.9 Transparency color keying 2456
46.5.10 Upscaler 2456
46.5.11 Display composition - windowing 2457
46.5.12 Display composition - blending 2458
46.5.13 Gamma correction 2460
46.5.14 YUV output conversion 2460
46.5.15 Dithering 2461
46.5.16 CRC hashing 2461
46.5.17 Display timing 2462
46.5.18 Output interface polarity 2464
46.5.19 Shadow registers 2464
46.5.20 General control 2466
46.5.21 Provision for a secure layer 2466
46.6 LTDC interrupts 2467

46.7 LTDC programming procedure 2468
46.8 LTDC registers 2469
46.8.1 LTDC identification register (LTDC_IDR) 2469
46.8.2 LDTC layer count register (LTDC_LCR) 2469
46.8.3 LTDC synchronization size configuration register (LTDC_SSCR) . . . 2470
46.8.4 LTDC back porch configuration register (LTDC_BPCR) 2470
46.8.5 LTDC active width configuration register (LTDC_AWCR) 2471
46.8.6 LTDC total width configuration register (LTDC_TWCR) 2471
46.8.7 LTDC global control register (LTDC_GCR) 2472
46.8.8 LTDC global configuration 1 register (LTDC_GC1R) 2474
46.8.9 LTDC global configuration 2 register (LTDC_GC2R) 2475
46.8.10 LTDC shadow reload configuration register (LTDC_SRCR) 2476
46.8.11 LTDC gamma correction configuration register (LTDC_GCCR) 2476
46.8.12 LTDC background color configuration register (LTDC_BCCR) 2477
46.8.13 LTDC interrupt enable register (LTDC_IER) 2477
46.8.14 LTDC interrupt status register (LTDC_ISR) 2478
46.8.15 LTDC interrupt clear register (LTDC_ICR) 2479
46.8.16 LTDC line interrupt position configuration register (LTDC_LIPCR) . . 2480
46.8.17 LTDC current position status register (LTDC_CPSR) 2480
46.8.18 LTDC current display status register (LTDC_CDSR) 2481
46.8.19 LTDC external display control register (LTDC_EDCR) 2481
46.8.20 LTDC interrupt enable register 2 (LTDC_IER2) 2482
46.8.21 LTDC interrupt status register 2 (LTDC_ISR2) 2483
46.8.22 LTDC interrupt clear register 2 (LTDC_ICR2) 2484
46.8.23 LTDC line interrupt position configuration register 2 (LTDC_LIPCR2) 2485
46.8.24 LTDC expected CRC register (LTDC_ECRCR) 2485
46.8.25 LTDC computed CRC register (LTDC_CCRCR) 2485
46.8.26 LTDC FIFO underrun threshold register (LTDC_FUTR) 2486
46.8.27 LTDC layer x configuration 0 register (LTDC_LxC0R) 2486
46.8.28 LTDC layer x configuration 1 register (LTDC_LxC1R) 2487
46.8.29 LTDC layer x reload control register (LTDC_LxRCR) 2488
46.8.30 LTDC layer x control register (LTDC_LxCR) 2489
46.8.31 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR) 2490
46.8.32 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR) 2490
46.8.33 LTDC layer x color keying configuration register (LTDC_LxCKCR) 2491

46.8.34 LTDC layer x pixel format configuration register (LTDC_LxPFCR) 2491
46.8.35 LTDC layer x constant alpha configuration register (LTDC_LxCACR) 2492
46.8.36 LTDC layer x default color configuration register (LTDC_LxDCCR) 2493
46.8.37 LTDC layer x blending factors configuration register (LTDC_LxBFCR) 2493
46.8.38 LTDC layer x burst length configuration register (LTDC_LxBLCR) 2494
46.8.39 LTDC layer x planar configuration register (LTDC_LxPCR) 2495
46.8.40 LTDC layer x color frame buffer address register (LTDC_LxCFBAR) 2496
46.8.41 LTDC layer x color frame buffer length register (LTDC_LxCFBLR) 2496
46.8.42 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR) 2497
46.8.43 LTDC layer x auxiliary frame buffer address 0 register (LTDC_LxAFBA0R) 2497
46.8.44 LTDC layer x auxiliary frame buffer address 1 register (LTDC_LxAFBA1R) 2498
46.8.45 LTDC layer x auxiliary frame buffer length register (LTDC_LxAFBLR) 2498
46.8.46 LTDC layer x auxiliary frame buffer line number register (LTDC_LxAFBLNR) 2499
46.8.47 LTDC layer x CLUT write register (LTDC_LxCLUTWR) 2499
46.8.48 LTDC layer x scaler input size register (LTDC_LxSISR) 2500
46.8.49 LTDC layer x scaler output size register (LTDC_LxSOSR) 2500
46.8.50 LTDC layer x scaler vertical scaling factor register (LTDC_LxSVSFR) 2501
46.8.51 LTDC layer x scaler vertical scaling phase register (LTDC_LxSVSPR) 2501
46.8.52 LTDC layer x scaler horizontal scaling factor register (LTDC_LxSHSFR) 2502
46.8.53 LTDC layer x scaler horizontal scaling phase register (LTDC_LxSHSPR) 2502
46.8.54 LTDC layer x conversion YCbCr RGB 0 register (LTDC_LxCYR0R) . 2502
46.8.55 LTDC layer x conversion YCbCr RGB 1 register (LTDC_LxCYR1R) . 2503
46.8.56 LTDC layer x flexible pixel format 0 register (LTDC_LxFPF0R) 2503
46.8.57 LTDC layer x flexible pixel format 1 register (LTDC_LxFPF1R) 2504
46.8.58 LTDC register map 2504

47 True random number generator (RNG) 2512
47.1 RNG introduction 2512
47.2 RNG main features 2512
47.3 RNG functional description 2513
47.3.1 RNG block diagram 2513
47.3.2 RNG internal signals 2513
47.3.3 Random number generation 2514
47.3.4 RNG initialization 2516
47.3.5 RNG operation 2517
47.3.6 RNG clocking 2519
47.3.7 Error management 2519
47.3.8 RNG low-power use 2520
47.4 RNG interrupts 2520
47.5 RNG processing time 2521
47.6 RNG entropy source validation 2521
47.6.1 Introduction 2521
47.6.2 Validation conditions 2521
47.7 RNG registers 2523
47.7.1 RNG control register (RNG_CR) 2523
47.7.2 RNG status register (RNG_SR) 2525
47.7.3 RNG data register (RNG_DR) 2526
47.7.4 RNG noise source control register (RNG_NSCR) 2526
47.7.5 RNG health test control register x (RNG_HTCRx) 2527
47.7.6 RNG hardware configuration register (RNG_HWCFGR) 2528
47.7.7 RNG version register (RNG_VERR) 2528
47.7.8 RNG identification register (RNG_IPIDR) 2529
47.7.9 RNG size identification register (RNG_SIDR) 2529
47.7.10 RNG register map 2529

48 Cryptographic processor (SAES) 2531
48.1 SAES introduction 2531
48.2 SAES main features 2532
48.3 SAES implementation 2533
48.4 SAES functional description 2534
48.4.1 SAES block diagram 2534
48.4.2 SAES internal signals 2534

48.4.3 SAES reset and clocks . . . 2535
48.4.4 SAES symmetric cipher implementation . . . 2535
48.4.5 SAES encryption or decryption typical usage . . . 2536
48.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication . . . 2538
48.4.7 SAES ciphertext stealing and data padding . . . 2539
48.4.8 SAES suspend and resume operations . . . 2539
48.4.9 SAES basic chaining modes (ECB, CBC) . . . 2540
48.4.10 SAES counter (CTR) mode . . . 2544
48.4.11 SAES Galois/counter mode (GCM) . . . 2546
48.4.12 SAES Galois message authentication code (GMAC) . . . 2551
48.4.13 SAES counter with CBC-MAC (CCM) . . . 2552
48.4.14 SAES basic operations with wrapped keys . . . 2557
48.4.15 Wrapped key usage outside the SAES peripheral . . . 2562
48.4.16 SAES data registers and data swapping . . . 2562
48.4.17 SAES key registers . . . 2565
48.4.18 SAES initialization vector registers . . . 2567
48.4.19 SAES error management . . . 2567
48.5 SAES interrupts . . . 2569
48.6 SAES DMA requests . . . 2570
48.6.1 Data input using DMA . . . 2570
48.6.2 Data output using DMA . . . 2570
48.6.3 Stopping DMA transfers . . . 2570
48.7 SAES processing latency . . . 2570
48.8 SAES registers . . . 2571
48.8.1 SAES control register (SAES_CR) . . . 2571
48.8.2 SAES status register (SAES_SR) . . . 2575
48.8.3 SAES data input register (SAES_DINR) . . . 2576
48.8.4 SAES data output register (SAES_DOUTR) . . . 2577
48.8.5 SAES key register 0 (SAES_KEYR0) . . . 2577
48.8.6 SAES key register 1 (SAES_KEYR1) . . . 2578
48.8.7 SAES key register 2 (SAES_KEYR2) . . . 2578
48.8.8 SAES key register 3 (SAES_KEYR3) . . . 2578
48.8.9 SAES key register 4 (SAES_KEYR4) . . . 2579
48.8.10 SAES key register 5 (SAES_KEYR5) . . . 2579
48.8.11 SAES key register 6 (SAES_KEYR6) . . . 2579
48.8.12 SAES key register 7 (SAES_KEYR7) . . . 2580

48.8.13 SAES initialization vector register 0 (SAES_IVR0) 2580
48.8.14 SAES initialization vector register 1 (SAES_IVR1) 2580
48.8.15 SAES initialization vector register 2 (SAES_IVR2) 2581
48.8.16 SAES initialization vector register 3 (SAES_IVR3) 2581
48.8.17 SAES context swap registers (SAES_SUSPRx) 2581
48.8.18 SAES interrupt enable register (SAES_IER) 2582
48.8.19 SAES interrupt status register (SAES_ISR) 2582
48.8.20 SAES interrupt clear register (SAES_ICR) 2584
48.8.21 SAES hardware configuration register (SAES_HWCFGR) 2585
48.8.22 SAES version register (SAES_VERR) 2585
48.8.23 SAES identification register (SAES_IPIDR) 2585
48.8.24 SAES size ID register (SAES_SIDR) 2586
48.8.25 SAES register map 2586

49 Cryptographic processor (CRYP) 2589

49.1 CRYP introduction 2589
49.2 CRYP main features 2589
49.3 CRYP implementation 2590
49.4 CRYP functional description 2591
49.4.1 CRYP block diagram 2591
49.4.2 CRYP internal signals 2592
49.4.3 CRYP reset and clocks 2592
49.4.4 CRYP symmetric cipher implementation 2592
49.4.5 CRYP encryption/ decryption typical usage 2593
49.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication 2596
49.4.7 CRYP ciphertext stealing and data padding 2597
49.4.8 CRYP suspend and resume operations 2597
49.4.9 CRYP DES/TDES basic chaining modes (ECB, CBC) 2598
49.4.10 CRYP AES basic chaining modes (ECB, CBC) 2601
49.4.11 CRYP counter mode (AES-CTR) 2605
49.4.12 CRYP AES Galois/counter mode (GCM) 2608
49.4.13 CRYP AES Galois message authentication code (GMAC) 2613
49.4.14 CRYP AES Counter with CBC-MAC (CCM) 2614
49.4.15 AES key sharing with secure AES co-processor 2619
49.4.16 CRYP data registers and data swapping 2620
49.4.17 CRYP key registers 2624

49.4.18 CRYP initialization vector registers 2625
49.4.19 CRYP error management 2625
49.5 CRYP interrupts 2626
49.6 CRYP DMA requests 2627
49.7 CRYP processing time 2628
49.8 CRYP registers 2629
49.8.1 CRYP control register (CRYP_CR) 2629
49.8.2 CRYP status register (CRYP_SR) 2632
49.8.3 CRYP data input register (CRYP_DINR) 2633
49.8.4 CRYP data output register (CRYP_DOUTR) 2634
49.8.5 CRYP DMA control register (CRYP_DMACR) 2634
49.8.6 CRYP interrupt mask set/clear register (CRYP_IMSCR) 2635
49.8.7 CRYP raw interrupt status register (CRYP_RISR) 2635
49.8.8 CRYP masked interrupt status register (CRYP_MISR) 2636
49.8.9 CRYP key register 0L (CRYP_K0LR) 2637
49.8.10 CRYP key register 0R (CRYP_K0RR) 2637
49.8.11 CRYP key register 1L (CRYP_K1LR) 2638
49.8.12 CRYP key register 1R (CRYP_K1RR) 2638
49.8.13 CRYP key register 2L (CRYP_K2LR) 2638
49.8.14 CRYP key register 2R (CRYP_K2RR) 2639
49.8.15 CRYP key register 3L (CRYP_K3LR) 2639
49.8.16 CRYP key register 3R (CRYP_K3RR) 2640
49.8.17 CRYP initialization vector register 0L (CRYP_IV0LR) 2640
49.8.18 CRYP initialization vector register 0R (CRYP_IV0RR) 2640
49.8.19 CRYP initialization vector register 1L (CRYP_IV1LR) 2641
49.8.20 CRYP initialization vector register 1R (CRYP_IV1RR) 2641
49.8.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR) 2642
49.8.22 CRYP context swap GCM registers (CRYP_CSGCMxR) 2642
49.8.23 CRYP hardware configuration register (CRYP_HWCFGR) 2643
49.8.24 CRYP version register (CRYP_VERR) 2643
49.8.25 CRYP identification (CRYP_IPIDR) 2644
49.8.26 CRYP size identification register (CRYP_SIDR) 2644
49.8.27 CRYP register map 2644

50 Hash processor (HASH) 2647

50.1 HASH introduction 2647

50.2 HASH main features . . . 2647
50.3 HASH implementation . . . 2648
50.4 HASH functional description . . . 2648
50.4.1 HASH block diagram . . . 2648
50.4.2 HASH internal signals . . . 2649
50.4.3 About secure hash algorithms . . . 2649
50.4.4 Message data feeding . . . 2650
50.4.5 Message digest computing . . . 2651
50.4.6 Message padding . . . 2654
50.4.7 HMAC operation . . . 2655
50.4.8 HASH suspend/resume operations . . . 2657
50.4.9 HASH DMA interface . . . 2659
50.4.10 HASH error management . . . 2660
50.4.11 HASH processing time . . . 2660
50.5 HASH interrupts . . . 2660
50.6 HASH registers . . . 2661
50.6.1 HASH control register (HASH_CR) . . . 2661
50.6.2 HASH data input register (HASH_DIN) . . . 2663
50.6.3 HASH start register (HASH_STR) . . . 2664
50.6.4 HASH digest registers . . . 2665
50.6.5 HASH interrupt enable register (HASH_IMR) . . . 2667
50.6.6 HASH status register (HASH_SR) . . . 2667
50.6.7 HASH context swap registers . . . 2669
50.6.8 HASH hardware configuration register (HASH_HWCFGR) . . . 2669
50.6.9 HASH version register (HASH_VERR) . . . 2670
50.6.10 HASH identification register (HASH_IPIDR) . . . 2670
50.6.11 HASH size identification register (HASH_SIDR) . . . 2671
50.6.12 HASH register map . . . 2671

51 On-the-fly decryption engine (OTFDEC) . . . 2673
51.1 OTFDEC introduction . . . 2673
51.2 OTFDEC main features . . . 2673
51.3 OTFDEC functional description . . . 2674
51.3.1 OTFDEC block diagram . . . 2674
51.3.2 OTFDEC internal signals . . . 2674
51.3.3 OTFDEC on-the-fly decryption . . . 2674

51.3.4 OTFDEC usage of AES in counter mode decryption 2676
51.3.5 Flow control management 2677
51.3.6 OTFDEC error management 2677
51.4 OTFDEC interrupts 2678
51.5 OTFDEC application information 2678
51.5.1 OTFDEC initialization process 2678
51.5.2 OTFDEC and power management 2679
51.5.3 Encrypting for OTFDEC 2680
51.5.4 OTFDEC key CRC source code 2681
51.6 OTFDEC registers 2682
51.6.1 OTFDEC control register (OTFDEC_CR) 2682
51.6.2 OTFDEC region x configuration register (OTFDEC_RxCFGR) 2682
51.6.3 OTFDEC region x start address register (OTFDEC_RxSTARTADDR) 2684
51.6.4 OTFDEC region x end address register (OTFDEC_RxENDADDR) . 2684
51.6.5 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) 2685
51.6.6 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) 2685
51.6.7 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) 2685
51.6.8 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) 2686
51.6.9 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) 2686
51.6.10 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) 2687
51.6.11 OTFDEC interrupt status register (OTFDEC_ISR) 2687
51.6.12 OTFDEC interrupt clear register (OTFDEC_ICR) 2688
51.6.13 OTFDEC interrupt enable register (OTFDEC_IER) 2689
51.6.14 OTFDEC hardware configuration register (OTFDEC_HWCFGR) . . . 2689
51.6.15 OTFDEC version register (OTFDEC_VERR) 2690
51.6.16 OTFDEC identification register (OTFDEC_IPIDR) 2690
51.6.17 OTFDEC size identification register (OTFDEC_SIDR) 2691
51.6.18 OTFDEC register map 2691

52 Public key accelerator (PKA) 2695
52.1 PKA introduction 2695
52.2 PKA main features 2695
52.3 PKA functional description 2696
52.3.1 PKA block diagram 2696
52.3.2 PKA internal signals 2696
52.3.3 PKA reset and clocks 2696

52.3.4 PKA public key acceleration . . . 2697
52.3.5 Typical applications for PKA . . . 2699
52.3.6 PKA procedure to perform an operation . . . 2701
52.3.7 PKA error management . . . 2702
52.4 PKA operating modes . . . 2703
52.4.1 Introduction . . . 2703
52.4.2 Montgomery parameter computation . . . 2704
52.4.3 Modular addition . . . 2704
52.4.4 Modular subtraction . . . 2705
52.4.5 Modular and Montgomery multiplication . . . 2705
52.4.6 Modular exponentiation . . . 2706
52.4.7 Modular inversion . . . 2708
52.4.8 Modular reduction . . . 2708
52.4.9 Arithmetic addition . . . 2709
52.4.10 Arithmetic subtraction . . . 2709
52.4.11 Arithmetic multiplication . . . 2710
52.4.12 Arithmetic comparison . . . 2710
52.4.13 RSA CRT exponentiation . . . 2710
52.4.14 Point on elliptic curve Fp check . . . 2711
52.4.15 ECC Fp scalar multiplication . . . 2712
52.4.16 ECDSA sign . . . 2713
52.4.17 ECDSA verification . . . 2715
52.4.18 ECC complete addition . . . 2716
52.4.19 ECC double base ladder . . . 2716
52.4.20 ECC projective to affine . . . 2717
52.5 Example of configurations and processing times . . . 2718
52.5.1 Supported elliptic curves . . . 2718
52.5.2 Computation times . . . 2720
52.6 PKA interrupts . . . 2722
52.7 PKA registers . . . 2723
52.7.1 PKA control register (PKA_CR) . . . 2723
52.7.2 PKA status register (PKA_SR) . . . 2725
52.7.3 PKA clear flag register (PKA_CLRFR) . . . 2726
52.7.4 PKA RAM . . . 2726
52.7.5 PKA hardware configuration register (PKA_HWCFGR) . . . 2727
52.7.6 PKA version register (PKA_VERR) . . . 2727
52.7.7 PKA identification register (PKA_IPIDR) . . . 2727

52.7.8 PKA size identification register (PKA_SIDR) 2728
52.7.9 PKA register map 2729

53 Advanced-control timers (TIM1/TIM8) 2730

53.1 TIM1/TIM8 introduction 2730
53.2 TIM1/TIM8 main features 2730
53.3 TIM1/TIM8 functional description 2731
53.3.1 Block diagram 2731
53.3.2 TIM1/TIM8 pins and internal signals 2732
53.3.3 Time-base unit 2736
53.3.4 Counter modes 2738
53.3.5 Repetition counter 2750
53.3.6 External trigger input 2751
53.3.7 Clock selection 2752
53.3.8 Capture/compare channels 2756
53.3.9 Input capture mode 2758
53.3.10 PWM input mode 2759
53.3.11 Forced output mode 2760
53.3.12 Output compare mode 2761
53.3.13 PWM mode 2762
53.3.14 Asymmetric PWM mode 2770
53.3.15 Combined PWM mode 2771
53.3.16 Combined 3-phase PWM mode 2772
53.3.17 Complementary outputs and dead-time insertion 2773
53.3.18 Using the break function 2776
53.3.19 Bidirectional break inputs 2782
53.3.20 Clearing the tim_ocxref signal on an external event 2783
53.3.21 6-step PWM generation 2784
53.3.22 One-pulse mode 2785
53.3.23 Retriggerable One-pulse mode 2787
53.3.24 Pulse on compare mode 2788
53.3.25 Encoder interface mode 2790
53.3.26 Direction bit output 2807
53.3.27 UIF bit remapping 2808
53.3.28 Timer input XOR function 2808
53.3.29 Interfacing with Hall sensors 2808
53.3.30 Timer synchronization 2810

53.3.31 ADC triggers . . . 2815
53.3.32 DMA burst mode . . . 2815
53.3.33 TIM1/TIM8 DMA requests . . . 2816
53.3.34 Debug mode . . . 2816
53.4 TIM1/TIM8 low-power modes . . . 2817
53.5 TIM1/TIM8 interrupts . . . 2817
53.6 TIM1/TIM8 registers . . . 2818
53.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8) . . . 2818
53.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8) . . . 2819
53.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) . . . 2823
53.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) . . . 2827
53.6.5 TIMx status register (TIMx_SR)(x = 1, 8) . . . 2828
53.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8) . . . 2831
53.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8) . . . 2832
53.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) . . . 2834
53.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8) . . . 2837
53.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) . . . 2838
53.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8) . . . 2841
53.6.12 TIMx counter (TIMx_CNT)(x = 1, 8) . . . 2845
53.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8) . . . 2845
53.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8) . . . 2846
53.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) . . . 2846
53.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) . . . 2847
53.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) . . . 2847
53.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) . . . 2848
53.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) . . . 2849
53.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) . . . 2850
53.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) . . . 2854
53.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) . . . 2855
53.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8) . . . 2856
53.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8) . . . 2857
53.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8) . . . 2858
53.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8) . . . 2859

53.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8) 2860
53.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8) 2863
53.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8) 2865
53.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) 2867
53.6.31 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 1, 8) 2867
53.6.32 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 1, 8) 2868
53.6.33 TIMx IP version register (TIMx_VERR)(x = 1, 8) 2869
53.6.34 TIMx IP identification register (TIMx_IPIDR)(x = 1, 8) 2870
53.6.35 TIMx size ID register (TIMx_SIDR)(x = 1, 8) 2870
53.6.36 TIMx register map 2870

54 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 2874

54.1 TIM2/TIM3/TIM4/TIM5 introduction 2874
54.2 TIM2/TIM3/TIM4/TIM5 main features 2874
54.3 TIM2/TIM3/TIM4/TIM5 implementation 2875
54.4 TIM2/TIM3/TIM4/TIM5 functional description 2876
54.4.1 Block diagram 2876
54.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 2877
54.4.3 Time-base unit 2880
54.4.4 Counter modes 2882
54.4.5 Clock selection 2894
54.4.6 Capture/compare channels 2898
54.4.7 Input capture mode 2900
54.4.8 PWM input mode 2901
54.4.9 Forced output mode 2902
54.4.10 Output compare mode 2902
54.4.11 PWM mode 2904
54.4.12 Asymmetric PWM mode 2912
54.4.13 Combined PWM mode 2913
54.4.14 Clearing the tim_ocxref signal on an external event 2914
54.4.15 One-pulse mode 2916
54.4.16 Retriggerable one-pulse mode 2917
54.4.17 Pulse on compare mode 2918
54.4.18 Encoder interface mode 2920
54.4.19 Direction bit output 2938

54.4.20 UIF bit remapping . . . 2939
54.4.21 Timer input XOR function . . . 2939
54.4.22 Timers and external trigger synchronization . . . 2939
54.4.23 Timer synchronization . . . 2943
54.4.24 ADC triggers . . . 2948
54.4.25 DMA burst mode . . . 2949
54.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests . . . 2950
54.4.27 Debug mode . . . 2950
54.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes . . . 2950
54.4.29 TIM2/TIM3/TIM4/TIM5 interrupts . . . 2951
54.5 TIM2/TIM3/TIM4/TIM5 registers . . . 2952
54.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) . . . 2952
54.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) . . . 2953
54.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) . . . 2955
54.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) . . . 2959
54.5.5 TIMx status register (TIMx_SR)(x = 2 to 5) . . . 2960
54.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5) . . . 2962
54.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5) . . . 2963
54.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) . . . 2965
54.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5) . . . 2967
54.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) . . . 2968
54.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5) . . . 2971
54.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5) . . . 2973
54.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5) . . . 2973
54.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 5) . . . 2974
54.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5) . . . 2974
54.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5) . . . 2975
54.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5) . . . 2976
54.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5) . . . 2977
54.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5) . . . 2978
54.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5) . . . 2979
54.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5) . . . 2980
54.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5) . . . 2981
54.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) . . . 2982
54.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) . . . 2983

54.5.25 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 2 to 5) . . . 2984
54.5.26 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 2 to 5) . . . 2985
54.5.27 TIMx IP version register (TIMx_VERR)(x = 2 to 5) . . . 2986
54.5.28 TIMx IP identification register (TIMx_IPIDR)(x = 2 to 5) . . . 2987
54.5.29 TIMx size ID register (TIMx_SIDR)(x = 2 to 5) . . . 2987
54.5.30 TIMx register map . . . 2987

55 Basic timers (TIM6/TIM7) . . . 2990

55.1 TIM6/TIM7 introduction . . . 2990
55.2 TIM6/TIM7 main features . . . 2990
55.3 TIM6/TIM7 functional description . . . 2991
55.3.1 TIM6/TIM7 block diagram . . . 2991
55.3.2 TIM6/TIM7 internal signals . . . 2991
55.3.3 TIM6/TIM7 clocks . . . 2992
55.3.4 Time-base unit . . . 2992
55.3.5 Counting mode . . . 2994
55.3.6 UIF bit remapping . . . 3001
55.3.7 ADC triggers . . . 3002
55.3.8 TIM6/TIM7 DMA requests . . . 3002
55.3.9 Debug mode . . . 3002
55.3.10 TIM6/TIM7 low-power modes . . . 3002
55.3.11 TIM6/TIM7 interrupts . . . 3002
55.4 TIM6/TIM7 registers . . . 3003
55.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) . . . 3003
55.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) . . . 3005
55.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) . . . 3005
55.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) . . . 3006
55.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) . . . 3006
55.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) . . . 3006
55.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) . . . 3007
55.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7) . . . 3007
55.4.9 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 6 to 7) . . . 3008
55.4.10 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 6 to 7) . . . 3008
55.4.11 TIMx IP version register (TIMx_VERR)(x = 6 to 7) . . . 3010

55.4.12 TIMx IP identification register (TIMx_IPIDR)(x = 6 to 7) 3010
55.4.13 TIMx size ID register (TIMx_SIDR)(x = 6 to 7) 3010
55.4.14 TIMx register map 3012

56 General-purpose timers (TIM10/TIM11/TIM12/TIM13/TIM14) 3014

56.1 TIM10/TIM11/TIM12/TIM13/TIM14 introduction 3014
56.2 TIM12 main features 3014
56.3 TIM10/TIM11/TIM13/TIM14 main features 3015
56.4 TIM10/TIM11/TIM12/TIM13/TIM14 functional description 3016
56.4.1 Block diagram 3016
56.4.2 TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals 3017
56.4.3 Time-base unit 3019
56.4.4 Counter modes 3021
56.4.5 Clock selection 3024
56.4.6 Capture/compare channels 3026
56.4.7 Input capture mode 3028
56.4.8 PWM input mode (TIM12 only) 3029
56.4.9 Forced output mode 3030
56.4.10 Output compare mode 3031
56.4.11 PWM mode 3032
56.4.12 Combined PWM mode (TIM12 only) 3037
56.4.13 One-pulse mode 3038
56.4.14 Retriggerable one pulse mode (TIM12 only) 3040
56.4.15 UIF bit remapping 3041
56.4.16 Timer input XOR function 3041
56.4.17 TIM12 external trigger synchronization 3041
56.4.18 Slave mode – combined reset + trigger mode 3044
56.4.19 Slave mode – combined reset + gated mode 3044
56.4.20 Timer synchronization (TIM12 only) 3044
56.4.21 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only) 3044
56.4.22 ADC triggers (TIM12 only) 3044
56.4.23 Debug mode 3044
56.5 TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes 3045
56.6 TIM10/TIM11/TIM12/TIM13/TIM14 interrupts 3045
56.7 TIM12 registers 3045
56.7.1 TIM12 control register 1 (TIM12_CR1) 3045

56.7.2 TIM12 control register 2 (TIM12_CR2) 3047
56.7.3 TIM12 slave mode control register (TIM12_SMCR) 3047
56.7.4 TIM12 interrupt enable register (TIM12_DIER) 3050
56.7.5 TIM12 status register (TIM12_SR) 3051
56.7.6 TIM12 event generation register (TIM12_EGR) 3052
56.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1) 3053
56.7.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) 3054
56.7.9 TIM12 capture/compare enable register (TIM12_CCER) 3057
56.7.10 TIM12 counter (TIM12_CNT) 3058
56.7.11 TIM12 prescaler (TIM12_PSC) 3059
56.7.12 TIM12 autoreload register (TIM12_ARR) 3059
56.7.13 TIM12 capture/compare register 1 (TIM12_CCR1) 3060
56.7.14 TIM12 capture/compare register 2 (TIM12_CCR2) 3060
56.7.15 TIM12 timer input selection register (TIM12_TISEL) 3061
56.7.16 TIM12 hardware configuration 2 register (TIM12_HWCFGR2) 3062
56.7.17 TIM12 hardware configuration 1 register (TIM12_HWCFGR1) 3062
56.7.18 TIM12 IP version register (TIM12_VERR) 3064
56.7.19 TIM12 IP identification register (TIM12_IPIDR) 3064
56.7.20 TIM12 size ID register (TIM12_SIDR) 3064
56.7.21 TIM12 register map 3066
56.8 TIM10/TIM11/TIM13/TIM14 registers 3068
56.8.1 TIMx control register 1 (TIMx_CR1)(x = 10, 11, 13, 14) 3068
56.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 10, 11, 13, 14) 3069
56.8.3 TIMx status register (TIMx_SR)(x = 10, 11, 13, 14) 3069
56.8.4 TIMx event generation register (TIMx_EGR)(x = 10, 11, 13, 14) 3070
56.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 10, 11, 13, 14) 3071
56.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 10, 11, 13, 14) 3072
56.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 10, 11, 13, 14) 3074
56.8.8 TIMx counter (TIMx_CNT)(x = 10, 11, 13, 14) 3075
56.8.9 TIMx prescaler (TIMx_PSC)(x = 10, 11, 13, 14) 3076
56.8.10 TIMx autoreload register (TIMx_ARR)(x = 10, 11, 13, 14) 3076
56.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 10, 11, 13, 14) . 3077
56.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 10, 11, 13, 14) 3077

56.8.13 TIMx hardware configuration 2 register (TIMx_HWCFGR2) (x = 10, 11, 13, 14) . . . 3078
56.8.14 TIMx hardware configuration 1 register (TIMx_HWCFGR1) (x = 10, 11, 13, 14) . . . 3079
56.8.15 TIMx IP version register (TIMx_VERR)(x = 10, 11, 13, 14) . . . 3080
56.8.16 TIMx IP identification register (TIMx_IPIDR)(x = 10, 11, 13, 14) . . . 3081
56.8.17 TIMx size ID register (TIMx_SIDR)(x = 10, 11, 13, 14) . . . 3081
56.8.18 TIM10/TIM11/TIM13/TIM14 register map . . . 3082

57 General purpose timers (TIM15/TIM16/TIM17) . . . 3084

57.1 TIM15/TIM16/TIM17 introduction . . . 3084
57.2 TIM15 main features . . . 3084
57.3 TIM16/TIM17 main features . . . 3085
57.4 TIM15/TIM16/TIM17 functional description . . . 3086
57.4.1 Block diagram . . . 3086
57.4.2 TIM15/TIM16/TIM17 pins and internal signals . . . 3087
57.4.3 Time-base unit . . . 3090
57.4.4 Counter modes . . . 3092
57.4.5 Repetition counter . . . 3096
57.4.6 Clock selection . . . 3097
57.4.7 Capture/compare channels . . . 3099
57.4.8 Input capture mode . . . 3101
57.4.9 PWM input mode (only for TIM15) . . . 3103
57.4.10 Forced output mode . . . 3104
57.4.11 Output compare mode . . . 3104
57.4.12 PWM mode . . . 3106
57.4.13 Combined PWM mode (TIM15 only) . . . 3111
57.4.14 Complementary outputs and dead-time insertion . . . 3112
57.4.15 Using the break function . . . 3115
57.4.16 Bidirectional break input . . . 3119
57.4.17 Clearing the tim_ocxref signal on an external event . . . 3120
57.4.18 6-step PWM generation . . . 3121
57.4.19 One-pulse mode . . . 3123
57.4.20 Retriggerable one pulse mode (TIM15 only) . . . 3124
57.4.21 UIF bit remapping . . . 3125
57.4.22 Timer input XOR function (TIM15 only) . . . 3125
57.4.23 External trigger synchronization (TIM15 only) . . . 3125

57.4.24 Slave mode – combined reset + trigger mode (TIM15 only) 3128
57.4.25 Slave mode – combined reset + gated mode (TIM15 only) 3128
57.4.26 Timer synchronization (TIM15 only) 3129
57.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 3129
57.4.28 ADC triggers (TIM15 only) 3129
57.4.29 DMA burst mode 3129
57.4.30 TIM15/TIM16/TIM17 DMA requests 3130
57.4.31 Debug mode 3131
57.5 TIM15/TIM16/TIM17 low-power modes 3131
57.6 TIM15/TIM16/TIM17 interrupts 3131
57.7 TIM15 registers 3132
57.7.1 TIM15 control register 1 (TIM15_CR1) 3132
57.7.2 TIM15 control register 2 (TIM15_CR2) 3133
57.7.3 TIM15 slave mode control register (TIM15_SMCR) 3135
57.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) 3137
57.7.5 TIM15 status register (TIM15_SR) 3138
57.7.6 TIM15 event generation register (TIM15_EGR) 3140
57.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) 3141
57.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 3143
57.7.9 TIM15 capture/compare enable register (TIM15_CCER) 3145
57.7.10 TIM15 counter (TIM15_CNT) 3148
57.7.11 TIM15 prescaler (TIM15_PSC) 3148
57.7.12 TIM15 autoreload register (TIM15_ARR) 3149
57.7.13 TIM15 repetition counter register (TIM15_RCR) 3149
57.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) 3150
57.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) 3151
57.7.16 TIM15 break and dead-time register (TIM15_BDTR) 3151
57.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) 3154
57.7.18 TIM15 input selection register (TIM15_TISEL) 3155
57.7.19 TIM15 alternate function register 1 (TIM15_AF1) 3156
57.7.20 TIM15 alternate function register 2 (TIM15_AF2) 3158
57.7.21 TIM15 DMA control register (TIM15_DCR) 3159
57.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) 3160
57.7.23 TIM15 hardware configuration 2 register (TIM15_HWCFGR2) 3160
57.7.24 TIM15 hardware configuration 1 register (TIM15_HWCFGR1) 3161
57.7.25 TIM15 IP version register (TIM15_VERR) 3162

57.7.26 TIM15 IP identification register (TIM15_IPIDR) 3163
57.7.27 TIM15 size ID register (TIM15_SIDR) 3163
57.7.28 TIM15 register map 3163
57.8 TIM16/TIM17 registers 3167
57.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) 3167
57.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) 3168
57.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 3169
57.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) 3170
57.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) 3171
57.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) 3172
57.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) 3173
57.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . 3175
57.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) 3178
57.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) 3178
57.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) 3179
57.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) 3179
57.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) 3180
57.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) 3181
57.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) 3184
57.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) 3185
57.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) 3185
57.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) 3188
57.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) 3188
57.8.20 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) 3189
57.8.21 TIMx hardware configuration 2 register (TIMx_HWCFGR2) (x = 16 to 17) 3190
57.8.22 TIMx hardware configuration 1 register (TIMx_HWCFGR1) (x = 16 to 17) 3191
57.8.23 TIMx IP version register (TIMx_VERR)(x = 16 to 17) 3192
57.8.24 TIMx IP identification register (TIMx_IPIDR)(x = 16 to 17) 3193
57.8.25 TIMx size ID register (TIMx_SIDR)(x = 16 to 17) 3193
57.8.26 TIM16/TIM17 register map 3194

58 Low-power timer (LPTIM) 3197

58.1 LPTIM introduction 3197

58.2 LPTIM main features 3197
58.3 LPTIM implementation 3198
58.4 LPTIM functional description 3199
58.4.1 LPTIM block diagram 3199
58.4.2 LPTIM pins and internal signals 3200
58.4.3 LPTIM input and trigger mapping 3202
58.4.4 LPTIM reset and clocks 3203
58.4.5 Glitch filter 3203
58.4.6 Prescaler 3204
58.4.7 Trigger multiplexer 3204
58.4.8 Operating mode 3205
58.4.9 Timeout function 3207
58.4.10 Waveform generation 3207
58.4.11 Register update 3208
58.4.12 Counter mode 3209
58.4.13 Timer enable 3210
58.4.14 Timer counter reset 3210
58.4.15 Encoder mode 3211
58.4.16 Repetition counter 3212
58.4.17 Capture/compare channels 3213
58.4.18 Input capture mode 3214
58.4.19 PWM mode 3216
58.4.20 Autonomous mode 3218
58.4.21 DMA requests 3219
58.4.22 Debug mode 3220
58.5 LPTIM low-power modes 3220
58.6 LPTIM interrupts 3220
58.7 LPTIM registers 3221
58.7.1 LPTIM5 interrupt and status register (LPTIM5_ISR) 3222
58.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 4) 3223
58.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 4) 3225
58.7.4 LPTIM5 interrupt clear register (LPTIM5_ICR) 3227
58.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 4) 3228
58.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 4) 3229

58.7.7 LPTIM5 interrupt enable register (LPTIM5_DIER) . . . 3230
58.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 4) . . . 3232
58.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 4) . . . 3233
58.7.10 LPTIM configuration register (LPTIM_CFGR) . . . 3235
58.7.11 LPTIM control register (LPTIM_CR) . . . 3238
58.7.12 LPTIM compare register 1 (LPTIM_CCR1) . . . 3239
58.7.13 LPTIM autoreload register (LPTIM_ARR) . . . 3240
58.7.14 LPTIM counter register (LPTIM_CNT) . . . 3240
58.7.15 LPTIM1 option register (LPTIM1_OR) . . . 3241
58.7.16 LPTIM configuration register 2 (LPTIM_CFGR2) . . . 3241
58.7.17 LPTIM repetition register (LPTIM_RCR) . . . 3242
58.7.18 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) . . . 3243
58.7.19 LPTIM compare register 2 (LPTIM_CCR2) . . . 3245
58.7.20 LPTIM peripheral hardware configuration register 2 (LPTIM_HWCFGR2) . . . 3246
58.7.21 LPTIM peripheral hardware configuration register 1 (LPTIM_HWCFGR1) . . . 3247
58.7.22 LPTIM peripheral version identification register (LPTIM_VERR) . . . 3247
58.7.23 LPTIM peripheral type identification register (LPTIM_PIDR) . . . 3248
58.7.24 LPTIM registers map size identification register (LPTIM_SIDR) . . . 3248
58.7.25 LPTIM register map . . . 3248

59 System timer generator (STGEN) . . . 3252

59.1 STGEN introduction . . . 3252
59.2 STGEN main features . . . 3252
59.3 STGEN functional description . . . 3252
59.3.1 STGEN block diagram . . . 3253
59.4 STGEN registers . . . 3254
59.4.1 STGENC control register (STGENC_CNTCR) . . . 3254
59.4.2 STGENC status register (STGENC_CNTSR) . . . 3254
59.4.3 STGENC value lower register (STGENC_CNTCVL) . . . 3255
59.4.4 STGENC value upper register (STGENC_CNTCVU) . . . 3255
59.4.5 STGENC base frequency register (STGENC_CNTFID0) . . . 3256
59.4.6 STGENC peripheral ID4 register (STGENC_PIDR4) . . . 3256
59.4.7 STGENC peripheral ID5 register (STGENC_PIDR5) . . . 3257
59.4.8 STGENC peripheral ID6 register (STGENC_PIDR6) . . . 3257

59.4.9 STGENC peripheral ID7 register (STGENC_PIDR7) 3257
59.4.10 STGENC peripheral ID0 register (STGENC_PIDR0) 3258
59.4.11 STGENC peripheral ID1 register (STGENC_PIDR1) 3258
59.4.12 STGENC peripheral ID2 register (STGENC_PIDR2) 3258
59.4.13 STGENC peripheral ID3 register (STGENC_PIDR3) 3259
59.4.14 STGENC component ID0 register (STGENC_CIDR0) 3259
59.4.15 STGENC component ID1 register (STGENC_CIDR1) 3260
59.4.16 STGENC component ID2 register (STGENC_CIDR2) 3260
59.4.17 STGENC component ID3 register (STGENC_CIDR3) 3261
59.4.18 STGENR value lower register (STGENR_CNTCVL) 3261
59.4.19 STGENR value upper register (STGENR_CNTCVU) 3261
59.4.20 STGENR peripheral ID4 register (STGENR_PIDR4) 3262
59.4.21 STGENR peripheral ID5 register (STGENR_PIDR5) 3262
59.4.22 STGENR peripheral ID6 register (STGENR_PIDR6) 3262
59.4.23 STGENR peripheral ID7 register (STGENR_PIDR7) 3263
59.4.24 STGENR peripheral ID0 register (STGENR_PIDR0) 3263
59.4.25 STGENR peripheral ID1 register (STGENR_PIDR1) 3263
59.4.26 STGENR peripheral ID2 register (STGENR_PIDR2) 3264
59.4.27 STGENR peripheral ID3 register (STGENR_PIDR3) 3264
59.4.28 STGENR component ID0 register (STGENR_CIDR0) 3265
59.4.29 STGENR component ID1 register (STGENR_CIDR1) 3265
59.4.30 STGENR component ID2 register (STGENR_CIDR2) 3265
59.4.31 STGENR component ID3 register (STGENR_CIDR3) 3266
59.4.32 STGEN register map 3266

60 Watchdog overview 3269

60.1 Watchdog main features 3269
60.2 Watchdog functional description 3269

61 Independent watchdog (IWDG) 3271

61.1 IWDG introduction 3271
61.2 IWDG main features 3271
61.3 IWDG implementation 3271
61.4 IWDG functional description 3272
61.4.1 IWDG block diagram 3272
61.4.2 IWDG internal signals 3273
61.4.3 Software and hardware watchdog modes 3273

61.4.4 Window option . . . 3274
61.4.5 Debug . . . 3277
61.4.6 Register access protection . . . 3277
61.5 IWDG low power modes . . . 3278
61.6 IWDG interrupts . . . 3278
61.7 IWDG registers . . . 3280
61.7.1 IWDG key register (IWDG_KR) . . . 3281
61.7.2 IWDG prescaler register (IWDG_PR) . . . 3281
61.7.3 IWDG reload register (IWDG_RLR) . . . 3282
61.7.4 IWDG status register (IWDG_SR) . . . 3282
61.7.5 IWDG window register (IWDG_WINR) . . . 3284
61.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) . . . 3284
61.7.7 IWDG interrupt clear register (IWDG_ICR) . . . 3285
61.7.8 IWDG hardware configuration register (IWDG_HWCFGR) . . . 3285
61.7.9 IWDG version register (IWDG_VERR) . . . 3286
61.7.10 IWDG identification register (IWDG_IDR) . . . 3287
61.7.11 IWDG size identification register (IWDG_SIDR) . . . 3287
61.7.12 IWDG register map . . . 3288

62 System window watchdog (WWDG) . . . 3289
62.1 WWDG introduction . . . 3289
62.2 WWDG main features . . . 3289
62.3 WWDG implementation . . . 3289
62.4 WWDG functional description . . . 3290
62.4.1 WWDG block diagram . . . 3290
62.4.2 WWDG internal signals . . . 3290
62.4.3 Enabling the watchdog . . . 3291
62.4.4 Controlling the down-counter . . . 3291
62.4.5 How to program the watchdog timeout . . . 3291
62.4.6 Debug mode . . . 3292
62.5 WWDG interrupts . . . 3293
62.6 WWDG registers . . . 3293
62.6.1 WWDG control register (WWDG_CR) . . . 3293
62.6.2 WWDG configuration register (WWDG_CFR) . . . 3294
62.6.3 WWDG status register (WWDG_SR) . . . 3294
62.6.4 WWDG hardware configuration register (WWDG_HWCFGR) . . . 3295

62.6.5 WWDG version register (WWDG_VERR) . . . 3295
62.6.6 WWDG identification register (WWDG_IPIDR) . . . 3295
62.6.7 WWDG size identification register (WWDG_SIDR) . . . 3296
62.6.8 WWDG register map . . . 3296

63 Real-time clock (RTC) . . . 3298
63.1 RTC introduction . . . 3298
63.2 RTC main features . . . 3298
63.3 RTC functional description . . . 3299
63.3.1 RTC block diagram . . . 3299
63.3.2 RTC pins and internal signals . . . 3301
63.3.3 GPIOs controlled by the RTC and TAMP . . . 3302
63.3.4 RTC secure protection modes . . . 3306
63.3.5 RTC privilege protection modes . . . 3307
63.3.6 RTC compartment-based isolation . . . 3309
63.3.7 Clock and prescalers . . . 3310
63.3.8 Real-time clock and calendar . . . 3311
63.3.9 Calendar ultra-low power mode . . . 3312
63.3.10 Programmable alarms . . . 3312
63.3.11 Periodic auto-wake-up . . . 3313
63.3.12 RTC initialization and configuration . . . 3313
63.3.13 Reading the calendar . . . 3316
63.3.14 Resetting the RTC . . . 3317
63.3.15 RTC synchronization . . . 3318
63.3.16 RTC reference clock detection . . . 3318
63.3.17 RTC smooth digital calibration . . . 3319
63.3.18 Timestamp function . . . 3321
63.3.19 Calibration clock output . . . 3322
63.3.20 Tamper and alarm output . . . 3322
63.4 RTC low-power modes . . . 3323
63.5 RTC interrupts . . . 3323
63.6 RTC registers . . . 3325
63.6.1 RTC time register (RTC_TR) . . . 3326
63.6.2 RTC date register (RTC_DR) . . . 3327
63.6.3 RTC subsecond register (RTC_SSR) . . . 3328
63.6.4 RTC initialization control and status register (RTC_ICSR) . . . 3329

63.6.5 RTC prescaler register (RTC_PRER) . . . 3331
63.6.6 RTC wake-up timer register (RTC_WUTR) . . . 3332
63.6.7 RTC control register (RTC_CR) . . . 3332
63.6.8 RTC privilege mode control register (RTC_PRIVCFGR) . . . 3336
63.6.9 RTC secure configuration register (RTC_SECCFGR) . . . 3338
63.6.10 RTC write protection register (RTC_WPR) . . . 3340
63.6.11 RTC calibration register (RTC_CALR) . . . 3340
63.6.12 RTC shift control register (RTC_SHIFTR) . . . 3342
63.6.13 RTC timestamp time register (RTC_TSTR) . . . 3343
63.6.14 RTC timestamp date register (RTC_TSDR) . . . 3344
63.6.15 RTC timestamp subsecond register (RTC_TSSSR) . . . 3345
63.6.16 RTC alarm A register (RTC_ALRMAR) . . . 3346
63.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) . . . 3347
63.6.18 RTC alarm B register (RTC_ALRMBR) . . . 3348
63.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) . . . 3350
63.6.20 RTC status register (RTC_SR) . . . 3351
63.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) . . . 3352
63.6.22 RTC secure masked interrupt status register (RTC_SMISR) . . . 3353
63.6.23 RTC status clear register (RTC_SCR) . . . 3355
63.6.24 RTC option register (RTC_OR) . . . 3356
63.6.25 RTC alarm A binary mode register (RTC_ALRABINR) . . . 3356
63.6.26 RTC alarm B binary mode register (RTC_ALRBBINR) . . . 3357
63.6.27 RTC Resource x CID configuration register (RTC_RxCIDCFGR) . . . 3357
63.6.28 RTC hardware configuration register 2 (RTC_HWCFGR2) . . . 3358
63.6.29 RTC hardware configuration register (RTC_HWCFGR1) . . . 3359
63.6.30 RTC version register (RTC_VERR) . . . 3360
63.6.31 RTC identification register (RTC_IPIDR) . . . 3360
63.6.32 RTC size identification register (RTC_SIDR) . . . 3360
63.6.33 RTC register map . . . 3361

64 Tamper and backup registers (TAMP) . . . 3364

64.1 TAMP introduction . . . 3364
64.2 TAMP main features . . . 3365
64.3 TAMP functional description . . . 3366
64.3.1 TAMP block diagram . . . 3366
64.3.2 TAMP pins and internal signals . . . 3367
64.3.3 GPIOs controlled by the RTC and TAMP . . . 3370

64.3.4 TAMP register write protection 3370
64.3.5 TAMP secure protection modes 3370
64.3.6 Backup registers protection zones 3371
64.3.7 TAMP privilege protection modes 3371
64.3.8 TAMP compartment-based isolation 3372
64.3.9 Boot hardware key (BHK) 3375
64.3.10 Tamper detection 3376
64.3.11 TAMP backup registers and other device secrets erase 3376
64.3.12 Tamper detection configuration and initialization 3378
64.4 TAMP low-power modes 3385
64.5 TAMP interrupts 3385
64.6 TAMP registers 3386
64.6.1 TAMP control register 1 (TAMP_CR1) 3386
64.6.2 TAMP control register 2 (TAMP_CR2) 3389
64.6.3 TAMP control register 3 (TAMP_CR3) 3391
64.6.4 TAMP filter control register (TAMP_FLTCR) 3393
64.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) 3394
64.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) 3397
64.6.7 TAMP active tamper output register (TAMP_ATOR) 3397
64.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) 3398
64.6.9 TAMP secure configuration register (TAMP_SECCFGR) 3401
64.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 3403
64.6.11 TAMP interrupt enable register (TAMP_IER) 3404
64.6.12 TAMP status register (TAMP_SR) 3406
64.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . 3408
64.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) 3409
64.6.15 TAMP status clear register (TAMP_SCR) 3411
64.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) 3414
64.6.17 TAMP monotonic counter 2 register (TAMP_COUNT2R) 3415
64.6.18 TAMP option register (TAMP_OR) 3415
64.6.19 TAMP resources protection configuration register (TAMP_RPCFGR) 3416
64.6.20 TAMP Backup registers RIF register 1(TAMP_BKPRIFR1) 3417
64.6.21 TAMP Backup registers RIF register 2(TAMP_BKPRIFR2) 3417
64.6.22 TAMP Backup registers RIF register 3(TAMP_BKPRIFR3) 3418
64.6.23 TAMP Resource x CID configuration register (TAMP_RxCIDCFGR) 3418
64.6.24 TAMP backup x register (TAMP_BKPxR) 3419
64.6.25 TAMP hardware configuration register 2 (TAMP_HWCFGR2) 3419

64.6.26 TAMP hardware configuration register 1 (TAMP_HWCFGR1) 3420
64.6.27 TAMP version register (TAMP_VERR) 3421
64.6.28 TAMP identification register (TAMP_IPIDR) 3421
64.6.29 TAMP size identification register (TAMP_SIDR) 3422
64.6.30 TAMP register map 3423

65 Inter-integrated circuit interface (I2C) 3426

65.1 I2C introduction 3426
65.2 I2C main features 3426
65.3 I2C implementation 3427
65.4 I2C functional description 3427
65.4.1 I2C block diagram 3428
65.4.2 I2C pins and internal signals 3428
65.4.3 I2C clock requirements 3431
65.4.4 I2C mode selection 3431
65.4.5 I2C initialization 3432
65.4.6 I2C reset 3436
65.4.7 I2C data transfer 3437
65.4.8 I2C target mode 3439
65.4.9 I2C controller mode 3448
65.4.10 I2C_TIMINGR register configuration examples 3459
65.4.11 SMBus specific features 3461
65.4.12 SMBus initialization 3463
65.4.13 SMBus I2C_TIMEOUTR register configuration examples 3465
65.4.14 SMBus target mode 3466
65.4.15 SMBus controller mode 3469
65.4.16 Autonomous mode 3472
65.4.17 Error conditions 3474
65.5 I2C in low-power modes 3475
65.6 I2C interrupts 3476
65.7 I2C DMA requests 3477
65.7.1 Transmission using DMA 3477
65.7.2 Reception using DMA 3477
65.7.3 Controller event control using DMA 3477
65.8 I2C debug modes 3478
65.9 I2C registers 3478

65.9.1 I2C control register 1 (I2C_CR1) . . . 3478
65.9.2 I2C control register 2 (I2C_CR2) . . . 3481
65.9.3 I2C own address 1 register (I2C_OAR1) . . . 3483
65.9.4 I2C own address 2 register (I2C_OAR2) . . . 3483
65.9.5 I2C timing register (I2C_TIMINGR) . . . 3485
65.9.6 I2C timeout register (I2C_TIMEOUTR) . . . 3486
65.9.7 I2C interrupt and status register (I2C_ISR) . . . 3487
65.9.8 I2C interrupt clear register (I2C_ICR) . . . 3489
65.9.9 I2C PEC register (I2C_PECR) . . . 3490
65.9.10 I2C receive data register (I2C_RXDR) . . . 3490
65.9.11 I2C transmit data register (I2C_TXDR) . . . 3491
65.9.12 I2C autonomous mode control register (I2C_AUTOCR) . . . 3491
65.9.13 I2C hardware configuration register (I2C_HWCFGR) . . . 3492
65.9.14 I2C version register (I2C_VERR) . . . 3493
65.9.15 I2C identification register (I2C_IPIDR) . . . 3493
65.9.16 I2C size identification register (I2C_SIDR) . . . 3493
65.9.17 I2C register map . . . 3494

66 Improved inter-integrated circuit (I3C) . . . 3496

66.1 I3C introduction . . . 3496
66.2 I3C main features . . . 3496
66.3 I3C implementation . . . 3498
66.3.1 I3C instantiation . . . 3498
66.3.2 I3C wake-up from low-power mode(s) . . . 3498
66.3.3 I3C autonomous mode with DMA in Stop mode(s) . . . 3498
66.3.4 I3C FIFOs . . . 3498
66.3.5 I3C triggers . . . 3499
66.3.6 I3C interrupt(s) . . . 3499
66.3.7 I3C MIPI® support . . . 3500
66.4 I3C block diagram . . . 3501
66.5 I3C pins and internal signals . . . 3501
66.6 I3C reset and clocks . . . 3502
66.6.1 I3C reset . . . 3502
66.6.2 I3C clocks and requirements . . . 3502
66.7 I3C peripheral state and programming . . . 3504
66.7.1 I3C peripheral state . . . 3504

66.7.2 I3C controller state and programming sequence . . . 3505
66.7.3 I3C target state and programming sequence . . . 3510
66.8 I3C registers and programming . . . 3514
66.8.1 I3C register set, as controller/target . . . 3514
66.8.2 I3C registers and fields use versus peripheral state, as controller . . . 3515
66.8.3 I3C registers and fields usage versus peripheral state, as target . . . 3518
66.9 I3C bus transfers and programming . . . 3520
66.9.1 I3C command set (CCCs), as controller/target . . . 3520
66.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller . . . 3524
66.9.3 I3C broadcast ENTDAA CCC transfer, as controller . . . 3526
66.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller . . . 3526
66.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target . . . 3528
66.9.6 I3C broadcast ENTDAA CCC transfer, as target . . . 3530
66.9.7 I3C broadcast DEFTGTS CCC transfer, as target . . . 3531
66.9.8 I3C broadcast DEFGRPA CCC transfer, as target . . . 3532
66.9.9 I3C direct GETSTATUS CCC response, as target . . . 3533
66.9.10 I3C private read/write transfer, as controller . . . 3534
66.9.11 I3C private read/write transfer, as target . . . 3535
66.9.12 Legacy I2C read/write transfer, as controller . . . 3536
66.9.13 I3C IBI transfer, as controller/target . . . 3537
66.9.14 I3C hot-join request transfer, as controller/target . . . 3539
66.9.15 I3C controller-role request transfer, as controller/target . . . 3540
66.10 I3C FIFOs management, as controller . . . 3541
66.10.1 C-FIFO management, as controller . . . 3541
66.10.2 TX-FIFO management, as controller . . . 3542
66.10.3 RX-FIFO management, as controller . . . 3545
66.10.4 S-FIFO management, as controller . . . 3547
66.11 I3C FIFOs management, as target . . . 3549
66.11.1 RX-FIFO management, as target . . . 3549
66.11.2 TX-FIFO management, as target . . . 3550
66.12 I3C error management . . . 3553
66.12.1 Controller error management . . . 3553
66.12.2 Target error management . . . 3555
66.13 I3C wake-up from low-power mode(s) . . . 3556
66.13.1 Wake-up from Stop . . . 3556

66.14 I3C in low-power modes 3559
66.15 I3C interrupts 3560
66.16 I3C registers 3560
66.16.1 I3C message control register (I3C_CR) 3560
66.16.2 I3C message control register [alternate] (I3C_CR) 3562
66.16.3 I3C configuration register (I3C_CFGR) 3564
66.16.4 I3C receive data byte register (I3C_RDR) 3569
66.16.5 I3C receive data word register (I3C_RDWR) 3570
66.16.6 I3C transmit data byte register (I3C_TDR) 3571
66.16.7 I3C transmit data word register (I3C_TDWR) 3572
66.16.8 I3C IBI payload data register (I3C_IBIDR) 3573
66.16.9 I3C target transmit configuration register (I3C_TGTTDR) 3574
66.16.10 I3C status register (I3C_SR) 3575
66.16.11 I3C status error register (I3C_SER) 3576
66.16.12 I3C received message register (I3C_RMR) 3578
66.16.13 I3C event register (I3C_EVR) 3579
66.16.14 I3C interrupt enable register (I3C_IER) 3583
66.16.15 I3C clear event register (I3C_CEVR) 3585
66.16.16 I3C masked interrupt status register (I3C_MISR) 3587
66.16.17 I3C own device characteristics register (I3C_DEVR0) 3588
66.16.18 I3C device x characteristics register (I3C_DEVRx) 3590
66.16.19 I3C maximum read length register (I3C_MAXRLR) 3592
66.16.20 I3C maximum write length register (I3C_MAXWLR) 3593
66.16.21 I3C timing register 0 (I3C_TIMINGR0) 3594
66.16.22 I3C timing register 1 (I3C_TIMINGR1) 3595
66.16.23 I3C timing register 2 (I3C_TIMINGR2) 3597
66.16.24 I3C bus characteristics register (I3C_BCR) 3598
66.16.25 I3C device characteristics register (I3C_DCR) 3599
66.16.26 I3C get capability register (I3C_GETCAPR) 3600
66.16.27 I3C controller-role capability register (I3C_CRCAPR) 3601
66.16.28 I3C get max data speed register (I3C_GETMXDSR) 3602
66.16.29 I3C extended provisioned ID register (I3C_EPIDR) 3604
66.16.30 I3C hardware configuration register (I3C_HWCFGR) 3605
66.16.31 I3C version register (I3C_VERR) 3605
66.16.32 I3C identification register (I3C_IPIDR) 3606
66.16.33 I3C size identification register (I3C_SIDR) 3606
66.16.34 I3C register map 3606

67 Universal synchronous/asynchronous receiver transmitter (USART/UART) 3610

67.1 USART introduction 3610
67.2 USART main features 3610
67.3 USART extended features 3611
67.4 USART implementation 3611
67.5 USART functional description 3613
67.5.1 USART block diagram 3613
67.5.2 USART pins and internal signals 3613
67.5.3 USART clocks 3616
67.5.4 USART character description 3616
67.5.5 USART FIFOs and thresholds 3618
67.5.6 USART transmitter 3618
67.5.7 USART receiver 3621
67.5.8 USART baud rate generation 3628
67.5.9 Tolerance of the USART receiver to clock deviation 3630
67.5.10 USART auto baud rate detection 3631
67.5.11 USART multiprocessor communication 3633
67.5.12 USART Modbus communication 3635
67.5.13 USART parity control 3636
67.5.14 USART LIN (local interconnection network) mode 3637
67.5.15 USART synchronous mode 3639
67.5.16 USART single-wire half-duplex communication 3643
67.5.17 USART receiver timeout 3643
67.5.18 USART smartcard mode 3644
67.5.19 USART IrDA SIR ENDEC block 3648
67.5.20 Continuous communication using USART and DMA 3651
67.5.21 RS232 hardware flow control and RS485 driver enable 3653
67.5.22 USART autonomous mode 3655
67.6 USART in low-power modes 3657
67.7 USART interrupts 3657
67.8 USART registers 3660
67.8.1 USART control register 1 (USART_CR1) 3660
67.8.2 USART control register 1 [alternate] (USART_CR1) 3664
67.8.3 USART control register 2 (USART_CR2) 3667
67.8.4 USART control register 3 (USART_CR3) 3671

67.8.5 USART control register 3 [alternate] (USART_CR3) 3675
67.8.6 USART baud rate register (USART_BRR) 3678
67.8.7 USART guard time and prescaler register (USART_GTPR) 3679
67.8.8 USART receiver timeout register (USART_RTOR) 3680
67.8.9 USART request register (USART_RQR) 3681
67.8.10 USART interrupt and status register (USART_ISR) 3682
67.8.11 USART interrupt and status register [alternate] (USART_ISR) 3688
67.8.12 USART interrupt flag clear register (USART_ICR) 3693
67.8.13 USART receive data register (USART_RDR) 3694
67.8.14 USART transmit data register (USART_TDR) 3695
67.8.15 USART prescaler register (USART_PRESC) 3695
67.8.16 USART autonomous mode control register (USART_AUTOCR) 3696
67.8.17 USART hardware configuration register 2 (USART_HWCFGR2) . . . 3697
67.8.18 USART hardware configuration register 1 (USART_HWCFGR1) . . . 3698
67.8.19 USART version register (USART_VERR) 3698
67.8.20 USART identification register (USART_IPIDR) 3699
67.8.21 USART size identification register (USART_SIDR) 3699
67.8.22 USART register map 3700

68 Low-power universal asynchronous receiver transmitter (LPUART) 3702

68.1 LPUART introduction 3702
68.2 LPUART main features 3702
68.3 LPUART implementation 3703
68.4 LPUART functional description 3705
68.4.1 LPUART block diagram 3705
68.4.2 LPUART pins and internal signals 3706
68.4.3 LPUART clocks 3707
68.4.4 LPUART character description 3708
68.4.5 LPUART FIFOs and thresholds 3710
68.4.6 LPUART transmitter 3710
68.4.7 LPUART receiver 3714
68.4.8 LPUART baud rate generation 3718
68.4.9 Tolerance of the LPUART receiver to clock deviation 3719
68.4.10 LPUART multiprocessor communication 3720
68.4.11 LPUART parity control 3722
68.4.12 LPUART single-wire half-duplex communication 3723

68.4.13 Continuous communication using DMA and LPUART 3723
68.4.14 RS232 hardware flow control 3726
68.4.15 LPUART autonomous mode 3727
68.5 LPUART in low-power modes 3729
68.6 LPUART interrupts 3730
68.7 LPUART registers 3731
68.7.1 LPUART control register 1 (LPUART_CR1) 3731
68.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 3734
68.7.3 LPUART control register 2 (LPUART_CR2) 3737
68.7.4 LPUART control register 3 (LPUART_CR3) 3739
68.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 3741
68.7.6 LPUART baud rate register (LPUART_BRR) 3743
68.7.7 LPUART request register (LPUART_RQR) 3743
68.7.8 LPUART interrupt and status register (LPUART_ISR) 3744
68.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 3749
68.7.10 LPUART interrupt flag clear register (LPUART_ICR) 3752
68.7.11 LPUART receive data register (LPUART_RDR) 3753
68.7.12 LPUART transmit data register (LPUART_TDR) 3753
68.7.13 LPUART prescaler register (LPUART_PRESC) 3754
68.7.14 LPUART autonomous mode control register (LPUART_AUTOCR) . . 3755
68.7.15 LPUART hardware configuration register 2 (LPUART_HWCFGR2) . 3756
68.7.16 LPUART hardware configuration register 1 (LPUART_HWCFGR1) . 3756
68.7.17 LPUART version register (LPUART_VERR) 3757
68.7.18 LPUART identification register (LPUART_IPIDR) 3757
68.7.19 LPUART size identification register (LPUART_SIDR) 3758
68.7.20 LPUART register map 3758

69 Serial peripheral interface (SPI) 3760
69.1 SPI introduction 3760
69.2 SPI main features 3760
69.3 SPI implementation 3761
69.4 SPI functional description 3762
69.4.1 SPI block diagram 3762
69.4.2 SPI pins and internal signals 3763
69.4.3 SPI communication general aspects 3765
69.4.4 Communications between one master and one slave 3765

69.4.5 Standard multislave communication . . . 3768
69.4.6 Multimaster communication . . . 3771
69.4.7 Slave select (NSS pin) management . . . 3772
69.4.8 Ready pin (RDY) management . . . 3776
69.4.9 Communication formats . . . 3776
69.4.10 Configuring the SPI . . . 3778
69.4.11 Enabling the SPI . . . 3779
69.4.12 SPI data transmission and reception procedures . . . 3780
69.4.13 Disabling the SPI . . . 3784
69.4.14 Communication using DMA (direct memory addressing) . . . 3785
69.4.15 Autonomous mode . . . 3786
69.5 SPI specific modes and control . . . 3788
69.5.1 TI mode . . . 3788
69.5.2 SPI error flags . . . 3789
69.5.3 CRC computation . . . 3792
69.6 SPI in low-power modes . . . 3793
69.7 SPI interrupts . . . 3793
69.8 I2S main features . . . 3795
69.9 I2S functional description . . . 3795
69.9.1 I2S general description . . . 3795
69.9.2 Pin sharing with SPI function . . . 3796
69.9.3 Bitfields usable in I2S/PCM mode . . . 3796
69.9.4 Slave and master modes . . . 3797
69.9.5 Supported audio protocols . . . 3797
69.9.6 Additional serial interface flexibility . . . 3803
69.9.7 Startup sequence . . . 3805
69.9.8 Stop sequence . . . 3807
69.9.9 Clock generator . . . 3808
69.9.10 Internal FIFOs . . . 3810
69.9.11 FIFO status flags . . . 3811
69.9.12 Handling of underrun situation . . . 3811
69.9.13 Handling of overrun situation . . . 3812
69.9.14 Frame error detection . . . 3813
69.9.15 DMA interface . . . 3815
69.9.16 Autonomous mode . . . 3815
69.9.17 Programing examples . . . 3815

69.10 I2S interrupts 3818
69.11 SPI/I2S registers 3818
69.11.1 SPI/I2S control register 1 (SPI_CR1) 3818
69.11.2 SPI/I2S control register 2 (SPI_CR2) 3820
69.11.3 SPI/I2S configuration register 1 (SPI_CFG1) 3821
69.11.4 SPI/I2S configuration register 2 (SPI_CFG2) 3824
69.11.5 SPI/I2S interrupt enable register (SPI_IER) 3826
69.11.6 SPI/I2S status register (SPI_SR) 3827
69.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) 3830
69.11.8 SPI/I2S autonomous mode control register (SPI_AUTOCR) 3831
69.11.9 SPI/I2S transmit data register (SPI_TXDR) 3831
69.11.10 SPI/I2S receive data register (SPI_RXDR) 3832
69.11.11 SPI/I2S polynomial register (SPI_CRCPOLY) 3833
69.11.12 SPI/I2S transmitter CRC register (SPI_TXCRC) 3833
69.11.13 SPI/I2S receiver CRC register (SPI_RXCRC) 3834
69.11.14 SPI/I2S underrun data register (SPI_UDRDR) 3835
69.11.15 SPI/I2S configuration register (SPI_I2SCFGR) 3835
69.11.16 SPI/I2S hardware configuration register (SPI_HWCFGR2) 3837
69.11.17 SPI/I2S hardware configuration register (SPI_HWCFGR1) 3838
69.11.18 SPI/I2S version register (SPI_VERR) 3839
69.11.19 SPI/I2S identification register (SPI_IPIDR) 3839
69.11.20 SPI/I2S size identification register (SPI_SIDR) 3839
69.11.21 SPI/I2S register map 3840

70 Serial audio interface (SAI) 3842
70.1 SAI introduction 3842
70.2 SAI main features 3842
70.3 SAI implementation 3843
70.4 SAI functional description 3844
70.4.1 SAI block diagram 3844
70.4.2 SAI pins and internal signals 3845
70.4.3 Main SAI modes 3846
70.4.4 SAI synchronization mode 3847
70.4.5 Audio data size 3848
70.4.6 Frame synchronization 3848
70.4.7 Slot configuration 3851

70.4.8 SAI clock generator . . . 3853
70.4.9 Internal FIFOs . . . 3856
70.4.10 PDM interface . . . 3858
70.4.11 AC’97 link controller . . . 3866
70.4.12 SPDIF output . . . 3868
70.4.13 Specific features . . . 3871
70.4.14 Error flags . . . 3875
70.4.15 Disabling the SAI . . . 3878
70.4.16 SAI DMA interface . . . 3878
70.5 SAI interrupts . . . 3879
70.6 SAI registers . . . 3881
70.6.1 SAI global configuration register (SAI_GCR) . . . 3881
70.6.2 SAI configuration register 1 (SAI_ACR1) . . . 3881
70.6.3 SAI configuration register 2 (SAI_ACR2) . . . 3884
70.6.4 SAI frame configuration register (SAI_AFRCR) . . . 3886
70.6.5 SAI slot register (SAI_ASLOTR) . . . 3887
70.6.6 SAI interrupt mask register (SAI_AIM) . . . 3888
70.6.7 SAI status register (SAI_ASR) . . . 3890
70.6.8 SAI clear flag register (SAI_ACLRFR) . . . 3892
70.6.9 SAI data register (SAI_ADR) . . . 3893
70.6.10 SAI configuration register 1 (SAI_BCR1) . . . 3893
70.6.11 SAI configuration register 2 (SAI_BCR2) . . . 3896
70.6.12 SAI frame configuration register (SAI_BFRCR) . . . 3898
70.6.13 SAI slot register (SAI_BSLOTR) . . . 3899
70.6.14 SAI interrupt mask register (SAI_BIM) . . . 3900
70.6.15 SAI status register (SAI_BSR) . . . 3901
70.6.16 SAI clear flag register (SAI_BCLRFR) . . . 3903
70.6.17 SAI data register (SAI_BDR) . . . 3904
70.6.18 SAI PDM control register (SAI_PDMCR) . . . 3905
70.6.19 SAI PDM delay register (SAI_PDMDLY) . . . 3906
70.6.20 SAI hardware configuration register (SAI_HWCFGR) . . . 3908
70.6.21 SAI version register (SAI_VERR) . . . 3909
70.6.22 SAI identification register (SAI_IPIDR) . . . 3909
70.6.23 SAI size identification register (SAI_SIDR) . . . 3910
70.6.24 SAI register map . . . 3910

71 SPDIF receiver interface (SPDIFRX) . . . 3912

71.1 SPDIFRX interface introduction 3912
71.2 SPDIFRX main features 3912
71.3 SPDIFRX functional description 3912
71.3.1 SPDIFRX pins and internal signals 3913
71.3.2 S/PDIF protocol (IEC-60958) 3914
71.3.3 SPDIFRX decoder (SPDIFRX_DC) 3916
71.3.4 SPDIFRX tolerance to clock deviation 3920
71.3.5 SPDIFRX synchronization 3920
71.3.6 SPDIFRX handling 3922
71.3.7 Data reception management 3924
71.3.8 Dedicated control flow 3926
71.3.9 Reception errors 3927
71.3.10 Clocking strategy 3929
71.3.11 Symbol clock generation 3929
71.3.12 DMA interface 3931
71.3.13 Interrupt generation 3932
71.3.14 Register protection 3933
71.4 Programming procedures 3933
71.4.1 Initialization phase 3934
71.4.2 Handling of interrupts coming from SPDIFRX 3935
71.4.3 Handling of interrupts coming from DMA 3935
71.5 SPDIFRX interface registers 3936
71.5.1 SPDIFRX control register (SPDIFRX_CR) 3936
71.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR) 3938
71.5.3 SPDIFRX status register (SPDIFRX_SR) 3939
71.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR) 3941
71.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR) 3942
71.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR) 3942
71.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR) 3943
71.5.8 SPDIFRX channel status register (SPDIFRX_CSR) 3944
71.5.9 SPDIFRX debug information register (SPDIFRX_DIR) 3944
71.5.10 SPDIFRX version register (SPDIFRX_VERR) 3945
71.5.11 SPDIFRX identification register (SPDIFRX_IPIDR) 3945
71.5.12 SPDIFRX size identification register (SPDIFRX_SIDR) 3946
71.5.13 SPDIFRX interface register map 3946

72 Controller area network with flexible data rate (FDCAN) 3948

72.1 FDCAN introduction 3948
72.2 FDCAN main features 3951
72.3 FDCAN implementation 3951
72.4 FDCAN functional description 3952
72.4.1 Operating modes 3953
72.4.2 Error management 3962
72.4.3 Message RAM 3962
72.4.4 FIFO acknowledge handling 3974
72.4.5 Bit timing 3975
72.4.6 Clock calibration on CAN 3976
72.4.7 Application 3980
72.4.8 TTCAN operations (FDCAN1 only) 3981
72.4.9 TTCAN configuration 3982
72.4.10 Message scheduling 3984
72.4.11 TTCAN gap control 3991
72.4.12 Stop watch 3992
72.4.13 Local time, cycle time, global time, and external clock synchronization 3992
72.4.14 TTCAN error level 3995
72.4.15 TTCAN message handling 3996
72.4.16 TTCAN interrupt and error handling 3999
72.4.17 Level 0 4000
72.4.18 Synchronization to external time schedule 4002
72.4.19 FDCAN Rx buffer and FIFO element 4003
72.4.20 FDCAN Tx buffer element 4005
72.4.21 FDCAN Tx event FIFO element 4007
72.4.22 FDCAN standard message ID filter element 4008
72.4.23 FDCAN extended message ID filter element 4010
72.4.24 FDCAN trigger memory element 4011
72.5 FDCAN registers 4013
72.5.1 FDCAN core release register (FDCAN_CREL) 4013
72.5.2 FDCAN Endian register (FDCAN_ENDN) 4013
72.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 4013
72.5.4 FDCAN test register (FDCAN_TEST) 4014
72.5.5 FDCAN RAM watchdog register (FDCAN_RWD) 4015
72.5.6 FDCAN CC control register (FDCAN_CCCR) 4016
72.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 4018

72.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 4019
72.5.9 FDCAN timestamp counter value register (FDCAN_TSCV) 4019
72.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 4020
72.5.11 FDCAN timeout counter value register (FDCAN_TOCV) 4021
72.5.12 FDCAN error counter register (FDCAN_ECR) 4021
72.5.13 FDCAN protocol status register (FDCAN_PSR) 4022
72.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 4024
72.5.15 FDCAN interrupt register (FDCAN_IR) 4024
72.5.16 FDCAN interrupt enable register (FDCAN_IE) 4027
72.5.17 FDCAN interrupt line select register (FDCAN_ILS) 4029
72.5.18 FDCAN interrupt line enable register (FDCAN_ILE) 4030
72.5.19 FDCAN global filter configuration register (FDCAN_GFC) 4031
72.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC) . . 4032
72.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC) . . 4032
72.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM) 4033
72.5.23 FDCAN high priority message status register (FDCAN_HPMS) 4034
72.5.24 FDCAN new data 1 register (FDCAN_NDAT1) 4034
72.5.25 FDCAN new data 2 register (FDCAN_NDAT2) 4035
72.5.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C) 4035
72.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 4036
72.5.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 4037
72.5.29 FDCAN Rx buffer configuration register (FDCAN_RXBC) 4037
72.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C) 4038
72.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 4039
72.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 4040
72.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC) 4040
72.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC) 4041
72.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 4042
72.5.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC) 4043
72.5.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 4043
72.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR) 4044
72.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 4045
72.5.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 4045
72.5.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 4046
72.5.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 4046

72.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 4047
72.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC) . . . 4047
72.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 4048
72.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 4049
72.5.47 FDCAN register map 4049
72.6 TTCAN registers 4053
72.6.1 FDCAN TT trigger memory configuration register (FDCAN_TTTMC) 4053
72.6.2 FDCAN TT reference message configuration register (FDCAN_TTRMC) 4053
72.6.3 FDCAN TT operation configuration register (FDCAN_TTOCF) 4054
72.6.4 FDCAN TT matrix limits register (FDCAN_TTMLM) 4056
72.6.5 FDCAN TUR configuration register (FDCAN_TURCF) 4057
72.6.6 FDCAN TT operation control register (FDCAN_TTOCN) 4058
72.6.7 FDCAN TT global time preset register (FDCAN_TTGTP) 4060
72.6.8 FDCAN TT time mark register (FDCAN_TTTMK) 4060
72.6.9 FDCAN TT interrupt register (FDCAN_TTIR) 4061
72.6.10 FDCAN TT interrupt enable register (FDCAN_TTIE) 4063
72.6.11 FDCAN TT interrupt line select register (FDCAN_TTILS) 4065
72.6.12 FDCAN TT operation status register (FDCAN_TTOST) 4066
72.6.13 FDCAN TUR numerator actual register (FDCAN_TURNA) 4068
72.6.14 FDCAN TT local and global time register (FDCAN_TTLGT) 4069
72.6.15 FDCAN TT cycle time and count register (FDCAN_TTCTC) 4069
72.6.16 FDCAN TT capture time register (FDCAN_TTCPT) 4070
72.6.17 FDCAN TT cycle sync mark register (FDCAN_TTCSM) 4070
72.6.18 FDCAN TT trigger select register (FDCAN_TTTS) 4071
72.6.19 FDCAN TT register map 4071
72.7 CCU registers 4073
72.7.1 Clock calibration unit core release register (FDCAN_CCU_CREL) . . 4073
72.7.2 Calibration configuration register (FDCAN_CCU_CCFG) 4073
72.7.3 Calibration status register (FDCAN_CCU_CSTAT) 4075
72.7.4 Calibration watchdog register (FDCAN_CCU_CWD) 4075
72.7.5 Clock calibration unit interrupt register (FDCAN_CCU_IR) 4076
72.7.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE) . . 4077
72.7.7 CCU register map 4077

73 USB subsystem (USBSS) 4079

73.1 Overview 4079
73.2 USB2 host high-speed port 4079
73.2.1 USB2 host high-speed port main features 4079
73.2.2 USB2 host high-speed port implementation 4081
73.2.3 USB2 host high-speed port functional description 4082
73.2.4 USB2 host high-speed host interrupts 4085
73.2.5 USB2PHY1 integration 4085
73.2.6 Connections to SYSCFG block 4087
73.3 USB2 OTG high-speed port 4089
73.3.1 USB2 OTG port main features 4089
73.3.2 USB2 OTG high-speed port functional description 4090
73.3.3 USB2 OTG high-speed port interrupts 4093
73.3.4 USB2PHY2 integration 4093
73.3.5 SYSCFG registers 4095
73.3.6 Connections to SYSCFG block 4096
73.4 Debug signals 4097

74 USB on-the-go high-speed (OTG) 4099
74.1 OTG introduction 4099
74.2 OTG main features 4100
74.2.1 General features 4100
74.2.2 Host-mode features 4101
74.2.3 Peripheral-mode features 4101
74.3 OTG implementation 4101
74.4 OTG functional description 4102
74.4.1 OTG block diagram 4102
74.4.2 OTG pin and internal signals 4102
74.4.3 OTG core 4103
74.4.4 High-speed OTG PHY connected to OTG 4103
74.4.5 Battery charging detection 4103
74.5 OTG dual role device (DRD) 4104
74.6 OTG as a USB peripheral 4104
74.6.1 Peripheral states 4105
74.6.2 Peripheral endpoints 4106
74.7 OTG as a USB host 4108
74.7.1 USB host states 4109

74.7.2 Host channels 4110
74.7.3 Host scheduler 4111
74.8 OTG SOF trigger 4112
74.8.1 Host SOFs 4113
74.8.2 Peripheral SOFs 4113
74.9 OTG low-power modes 4113
74.10 OTG Dynamic update of the OTG_HFIR register 4114
74.11 OTG data FIFOs 4115
74.11.1 Peripheral FIFO architecture 4115
74.11.2 Host FIFO architecture 4116
74.11.3 FIFO RAM allocation 4118
74.12 **OTG** interrupts 4119
74.13 **OTG** control and status registers 4121
74.13.1 CSR memory map 4121
74.14 **OTG** registers 4126
74.14.1 OTG control and status register (OTG_GOTGCTL) 4126
74.14.2 OTG interrupt register (OTG_GOTGINT) 4128
74.14.3 OTG AHB configuration register (OTG_GAHBCFG) 4129
74.14.4 OTG USB configuration register (OTG_GUSBCFG) 4130
74.14.5 OTG reset register (OTG_GRSTCTL) 4132
74.14.6 OTG core interrupt register [alternate] (OTG_GINTSTS) 4135
74.14.7 OTG core interrupt register [alternate] (OTG_GINTSTS) 4139
74.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK) 4144
74.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK) 4145
74.14.10 OTG receive status debug read register [alternate] (OTG_GRXSTSR) 4147
74.14.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR) 4148
74.14.12 OTG status read and pop registers (OTG_GRXSTSP) 4149
74.14.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP) . . 4150
74.14.14 OTG receive FIFO size register (OTG_GRXFSIZ) 4151
74.14.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ) 4152
74.14.16 Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0) 4152
74.14.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) 4153
74.14.18 OTG general core configuration register (OTG_GCCFG) 4154

74.14.19 OTG core ID register (OTG_CID) 4155
74.14.20 OTG core LPM configuration register (OTG_GLPMCFG) 4156
74.14.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ) 4160
74.14.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx) 4160
74.14.23 Host-mode registers 4160
74.14.24 OTG host configuration register (OTG_HCFG) 4161
74.14.25 OTG host frame interval register (OTG_HFIR) 4161
74.14.26 OTG host frame number/frame time remaining register (OTG_HFNUM) 4162
74.14.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS) 4163
74.14.28 OTG host all channels interrupt register (OTG_HAINT) 4164
74.14.29 OTG host all channels interrupt mask register (OTG_HAINTMSK) 4164
74.14.30 OTG host port control and status register (OTG_HPRT) 4165
74.14.31 OTG host channel x characteristics register (OTG_HCCHARx) 4167
74.14.32 OTG host channel x split control register (OTG_HCSPLTx) 4168
74.14.33 OTG host channel x interrupt register (OTG_HCINTx) 4169
74.14.34 OTG host channel x interrupt mask register (OTG_HCINTMSKx) . . 4170
74.14.35 OTG host channel x transfer size register (OTG_HCTSIZx) 4171
74.14.36 OTG host channel x DMA address register(OTG_HCDMAx) 4172
74.14.37 Device-mode registers 4172
74.14.38 OTG device configuration register (OTG_DCFG) 4172
74.14.39 OTG device control register (OTG_DCTL) 4174
74.14.40 OTG device status register (OTG_DSTS) 4176
74.14.41 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) 4177
74.14.42 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) 4178
74.14.43 OTG device all endpoints interrupt register (OTG_DAINT) 4179
74.14.44 OTG all endpoints interrupt mask register (OTG_DAINTMSK) 4180
74.14.45 OTG device threshold control register (OTG_DTHRCTL) 4180
74.14.46 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) 4181
74.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) 4182

74.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) 4184
74.14.49 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) 4186
74.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) 4187
74.14.51 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx) 4188
74.14.52 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) 4188
74.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) . 4189
74.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) 4189
74.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) . . 4191
74.14.56 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) 4193
74.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx) 4194
74.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) 4194
74.14.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) 4196
74.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) 4198
74.14.61 OTG power and clock gating control register (OTG_PCGCCTL) . . . 4199
74.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1) . 4200
74.14.63 OTG register map 4201
74.15 **OTG** programming model 4208
74.15.1 Core initialization 4208
74.15.2 Host initialization 4209
74.15.3 Device initialization 4210
74.15.4 DMA mode 4210
74.15.5 Host programming model 4210
74.15.6 Device programming model 4243
74.15.7 Worst case response time 4263
74.15.8 OTG programming model 4265

75 USB2 EHCI/OHCI high/full/low-speed host controller (USBH) 4266

75.1 Introduction 4266
75.2 USBH main features 4266
75.3 USBH functional description 4267

75.3.1 USBH block diagram 4267
75.3.2 Description of USBH packet buffer 4267
75.3.3 Description of USBH root hub 4267
75.4 USBH interrupts 4268
75.5 USBH registers 4268
75.5.1 USBH OHCI standard registers 4268
75.5.2 USBH OHCI implementation register #6 (USBH_OHCI_INSNREG06) 4269
75.5.3 USBH OHCI implementation register #7 (USBH_OHCI_INSNREG07) 4270
75.5.4 USBH EHCI standard registers 4270
75.5.5 USBH EHCI implementation register #1 (USBH_EHCI_INSNREG01) 4271
75.5.6 USBH EHCI implementation register #2 (USBH_EHCI_INSNREG02) 4272
75.5.7 USBH EHCI implementation register #3 (USBH_EHCI_INSNREG03) 4272
75.5.8 USBH EHCI implementation register #4 (USBH_EHCI_INSNREG04) 4274
75.5.9 USBH EHCI implementation register #5 (USBH_EHCI_INSNREG05) 4275
75.5.10 USBH EHCI implementation register #6 (USBH_EHCI_INSNREG06) 4276
75.5.11 USBH EHCI implementation register #7 (USBH_EHCI_INSNREG07) 4276
75.5.12 USBH register map 4277

76 Ethernet subsystem overview (ETHSS) 4281
76.1 Introduction 4281
76.2 ETHSS main features 4281
76.3 ETHSS functional description 4281
76.3.1 ETHSS block diagram 4281
76.3.2 ETHSS pins and internal signals 4282
76.3.3 ETHSS clocks 4285

77 Ethernet (ETH): gigabit media access control (GMAC) with DMA controller 4286
77.1 Ethernet introduction 4286
77.2 Ethernet main features 4286

77.2.1 Standard compliance 4286
77.2.2 MAC features 4286
77.2.3 Transaction layer (MTL) features 4288
77.2.4 DMA block features 4289
77.2.5 Bus interface features 4290
77.2.6 Audio and video features 4291
77.2.7 Time sensitive networking features 4291
77.2.8 Generic queuing features 4291
77.3 Ethernet pins and internal signals 4291
77.4 Ethernet architecture 4293
77.4.1 DMA controller 4295
77.4.2 MTL 4302
77.4.3 MAC 4302
77.4.4 Multiple channels and queues support 4307
77.5 Ethernet functional description: MAC 4319
77.5.1 Double VLAN processing 4319
77.5.2 Source address and VLAN insertion, replacement, or deletion 4320
77.5.3 Queue/channel-based VLAN tag insertion on Tx 4321
77.5.4 Packet filtering 4322
77.5.5 IEEE 1588 timestamp support 4334
77.5.6 Time sensitive networking (TSN) 4360
77.5.7 Checksum offload engine 4389
77.5.8 TCP segmentation offload 4395
77.5.9 IPv4 ARP offload 4401
77.5.10 Loopback 4402
77.5.11 Flow control 4403
77.5.12 MAC management counters 4407
77.5.13 Interrupts generated by the MAC 4410
77.5.14 MAC and MMC register descriptions 4410
77.6 Ethernet functional description: PHY interfaces 4411
77.6.1 Station management agent (SMA) 4411
77.6.2 Media independent interface (MII) 4418
77.6.3 Reduced media independent interface (RMII) 4419
77.6.4 Reduced gigabit media independent interface (RGMII) 4422
77.7 Ethernet low-power modes 4424
77.7.1 Low-power management 4424

77.7.2 Energy Efficient Ethernet (EEE) 4430
77.8 Ethernet interrupts 4437
77.8.1 DMA interrupts 4437
77.8.2 MTL interrupts 4438
77.8.3 MAC Interrupts 4439
77.9 Ethernet programming model 4440
77.9.1 DMA initialization 4440
77.9.2 MTL initialization 4442
77.9.3 MAC initialization 4442
77.9.4 Performing normal receive and transmit operation 4443
77.9.5 Stopping and starting transmission 4444
77.9.6 Programming guidelines for switching to new descriptor list in RxDMA 4444
77.9.7 Programming guidelines for switching the AXI clock frequency 4444
77.9.8 Programming guidelines for multichannel multiqueue operation 4445
77.9.9 Programming guidelines for GMII link state transitions 4447
77.9.10 Programming guidelines for IEEE 1588 timestamping 4448
77.9.11 Programming guidelines for PTP offload feature 4449
77.9.12 Programming guidelines for AV feature 4453
77.9.13 Programming guidelines for Energy Efficient Ethernet (EEE) 4455
77.9.14 Programming guidelines for flexible pulse-per-second (PPS) output 4456
77.9.15 Programming guidelines for IEEE 1588 auxiliary snapshot 4459
77.9.16 Programming guidelines for TSO 4459
77.9.17 Programming guidelines to perform VLAN filtering on the receiver . . 4460
77.9.18 Programming guidelines for extended VLAN filtering and routing on reception 4461
77.9.19 Programming sequence for queue/channel-based VLAN inclusion register 4461
77.9.20 Programming guidelines for EST 4462
77.9.21 Programming the launch time in time-based scheduling 4464
77.9.22 Enabling the frame preemption function 4464
77.10 Descriptors 4465
77.10.1 Descriptor overview 4465
77.10.2 Descriptor structure 4466
77.10.3 Transmit descriptor 4469
77.10.4 Receive descriptor 4482
77.10.5 Enhanced descriptor for time-based scheduling 4495

77.11 Ethernet registers 4499
77.11.1 Ethernet register maps 4499
77.11.2 Ethernet DMA registers 4499
77.11.3 Ethernet MTL registers 4540
77.11.4 Ethernet MAC and MMC registers 4574

78 Hardware debug port (HDP) 4737
78.1 HDP main features 4737
78.2 Functional description 4737
78.3 HDP registers 4739
78.3.1 HDP control register (HDP_CTRL) 4739
78.3.2 HDP multiplexer control register (HDP_MUX) 4739
78.3.3 HDP read back value register (HDP_VAL) 4740
78.3.4 HDP general purpose output set register (HDP_GPOSET) 4740
78.3.5 HDP general purpose output clear register (HDP_GPOCLR) 4740
78.3.6 HDP general purpose output value register (HDP_GPOVAL) 4741
78.3.7 HDP version register (HDP_VERR) 4741
78.3.8 HDP identification register (HDP_IPIDR) 4742
78.3.9 HDP size identification register (HDP_SIDR) 4742
78.3.10 HDP register map 4742

79 Software error robustness collector (SERC) 4744
79.1 Introduction 4744
79.2 SERF main features 4744
79.3 SERF functional description 4744
79.3.1 SERF block diagram 4744
79.3.2 SERF key internal signals 4746
79.3.3 SERF resets and clocks 4746
79.3.4 Functionality and programming model 4746
79.4 SERF low-power modes 4752
79.5 SERC interrupts 4752
79.6 SERC registers 4753
79.6.1 SERC interrupt enable register 0 (SERC_IER0) 4753
79.6.2 SERC interrupt enable register 1 (SERC_IER1) 4754
79.6.3 SERC interrupt enable register 2 (SERC_IER2) 4754
79.6.4 SERC interrupt enable register 3 (SERC_IER3) 4755

79.6.5 SERC interrupt enable register 4 (SERC_IER4) 4755
79.6.6 SERC interrupt enable register 5 (SERC_IER5) 4755
79.6.7 SERC interrupt status register 0 (SERC_ISR0) 4756
79.6.8 SERC interrupt status register 1 (SERC_ISR1) 4756
79.6.9 SERC interrupt status register 2 (SERC_ISR2) 4757
79.6.10 SERC interrupt status register 3 (SERC_ISR3) 4757
79.6.11 SERC interrupt status register 4 (SERC_ISR4) 4757
79.6.12 SERC interrupt status register 5 (SERC_ISR5) 4758
79.6.13 SERC interrupt clear register 0 (SERC_ICR0) 4758
79.6.14 SERC interrupt clear register 1 (SERC_ICR1) 4759
79.6.15 SERC interrupt clear register 2 (SERC_ICR2) 4759
79.6.16 SERC interrupt clear register 3 (SERC_ICR3) 4759
79.6.17 SERC interrupt clear register 4 (SERC_ICR4) 4760
79.6.18 SERC interrupt clear register 5 (SERC_ICR5) 4760
79.6.19 SERC enable (SERC_ENABLE) 4761
79.6.20 SERC hardware configuration register (SERC_HWCFGR) 4761
79.6.21 SERC version register (SERC_VERR) 4762
79.6.22 SERC identification register (SERC_IPIDR) 4762
79.6.23 SERC size identification register (SERC_SIDR) 4762
79.6.24 SERC register map 4763

80 Debug support (DBG) 4765

80.1 DBG introduction 4765
80.2 DBG use cases 4765
80.3 DBG block diagram 4767
80.4 DBG pins 4767
80.5 DBG power and clocking 4768
80.5.1 DBG power domains 4768
80.5.2 DBG clocks 4769
80.5.3 Debug and low-power modes 4770
80.5.4 Debug reset 4770
80.6 Security 4771
80.6.1 Authentication signals 4771
80.7 Chip-level TAP controller (CLTAPC) 4773
80.8 Serial-wire and JTAG debug port (SWJ-DP) 4773
80.8.1 JTAG debug port 4774

80.8.2 SWD debug port 4774
80.8.3 Access ports 4774
80.8.4 Authentication 4776
80.9 System debug features 4777
80.9.1 System ROM tables 4777
80.9.2 CoreSight global timestamp generator (TSGEN) 4779
80.9.3 Cross trigger interface (CTI) and matrix (CTM) 4781
80.9.4 Trace funnel (CSTF) 4788
80.9.5 Embedded trace FIFO (ETF) 4788
80.9.6 Embedded trace router (ETR) 4789
80.9.7 Trace port interface unit (TPIU) 4790
80.9.8 Replicator 4790
80.9.9 Serial-wire output (SWO) 4791
80.9.10 System trace macrocell (STM) 4791
80.9.11 Microcontroller debug unit (DBGMCU) 4799
80.10 Cortex-A35 debug features 4819
80.10.1 Cortex-A35 debug unit 4820
80.10.2 Cortex-A35 performance monitoring unit (PMU) 4820
80.10.3 Embedded trace macrocell (ETM) 4820
80.10.4 Cross trigger interface (CTI) 4821
80.11 Cortex-M33 debug features 4821
80.11.1 ROM tables 4821
80.11.2 Data watchpoint and trace unit (DWT) 4823
80.11.3 Instrumentation trace macrocell (ITM) 4823
80.11.4 Breakpoint unit (FPB) 4824
80.11.5 Embedded trace macrocell (ETM) 4824
80.11.6 Cross trigger interface (CTI) 4824
80.12 Debug authentication 4825
80.12.1 Main characteristics 4825
80.12.2 Debug authentication protocol principle 4825
80.12.3 Debug authentication protocol details 4826
80.13 DBGMCU authentication protocol status mailbox 4826
80.14 Debug versus TIL 4827
80.15 References 4827

81 Device electronic signature 4828

81.1 Unique device ID register (96 bits) (UID) 4828
81.2 Device part number (RPN) 4829
81.3 Device version 4830
81.4 Package data register (PKG) 4831

82 Important security notice 4832

83 Revision history 4833

List of tables

Table 1. STNoC routing of the DDR traffic . . . 173
Table 2. Connectivity matrix . . . 174
Table 3. STNoC master port main characteristics . . . 178
Table 4. Slave port main characteristics . . . 179
Table 5. MLAHB300 master port main characteristics . . . 181
Table 6. MLAHB300 target port main characteristics . . . 182
Table 7. MLAHB150 master port main characteristics . . . 182
Table 8. MLAHB150 slave ports main characteristics . . . 183
Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2) . . . 186
Table 10. Cortex-A35 subsystem configuration . . . 203
Table 11. Cortex-A35 subsystem reset sources . . . 204
Table 12. CA35SS SYSCFG register map and reset value . . . 215
Table 13. Mapping SSC registers to secure flag register . . . 218
Table 14. CA35SS SSC register map and reset value . . . 225
Table 15. Cortex-M33 reset sources . . . 230
Table 16. IDAU partitioning . . . 232
Table 17. Memory partitioning . . . 233
Table 18. Overview of RIF core components . . . 238
Table 19. Interrupt connection types . . . 242
Table 20. Access-type definitions for Arm TrustZone cores . . . 242
Table 21. RIF illegal access definitions (security) . . . 242
Table 22. RIF illegal access definitions (privilege) . . . 243
Table 23. RIF illegal access definitions (compartments) . . . 243
Table 24. BSEC internal input/output signals . . . 250
Table 25. BSEC initial status reporting structure . . . 255
Table 26. BSEC ad-hoc error/status reporting structure . . . 256
Table 27. BSEC states definition . . . 256
Table 28. Current HDPL coding . . . 259
Table 29. BSEC Next HDPL register usage . . . 260
Table 30. Debug and trace profiles . . . 260
Table 31. BSEC debug register legal values . . . 261
Table 32. BSEC register map and reset values . . . 284
Table 33. STM32MP21xx OTP mapping . . . 288
Table 34. STM32MP21xx OTP fuse description (lower OTP region) . . . 294
Table 35. STM32MP21xx OTP fuse description (mid OTP region) . . . 302
Table 36. STM32MP21xx OTP fuse description (upper OTP region) . . . 304
Table 37. RIMU resource assignment . . . 308
Table 38. Indexes for non-RIF-aware peripherals (RISUP) . . . 309
Table 39. Indexes for RIF-aware peripherals (with RCC control index) . . . 310
Table 40. RIMU internal input/output signal . . . 312
Table 41. RISUP internal input/output signals . . . 312
Table 42. DAP compartment ID allocation . . . 313
Table 43. RIFSC register map and reset values . . . 327
Table 44. RISAB resource assignment . . . 330
Table 45. RISAB internal input/output signals . . . 331
Table 46. RISAB register map and reset values . . . 345
Table 47. RISAF resource assignment . . . 348

Table 48. RISAF internal input/output signals . . . 349
Table 49. RISAF subregion security setup matrix . . . 351
Table 50. RISAF register map and reset values . . . 373
Table 51. IAC internal signals . . . 377
Table 52. IAC interrupt request . . . 378
Table 53. IAC register map and reset values . . . 382
Table 54. Frequency values for the DDRSS clocks versus data rate versus DDR JEDEC standard . . . 385
Table 55. List of observable signals sent directly to HDP . . . 389
Table 56. List of observable signals selectable and sent to HDP . . . 390
Table 57. Detailed AC swizzling register values . . . 398
Table 58. DDRSS register map and reset values . . . 402
Table 59. DDRCTRL register map and reset values . . . 509
Table 60. Function of DDRPHYC pins versus DDR protocol . . . 527
Table 61. DDRPHYC register map and reset values . . . 658
Table 62. Signals coming from DFI command-address decoder . . . 696
Table 63. Signals monitored at performance logging interface (provided by DDRCTRL) . . . 697
Table 64. Events generated by the DFI command address decoder . . . 698
Table 65. Filtering policy . . . 704
Table 66. DDRPERFM register map and reset values . . . 711
Table 67. Internal SRAM features . . . 714
Table 68. Internal SRAM hardware erase feature . . . 714
Table 69. Effect of low-power modes on RAMCFG . . . 716
Table 70. RAMCFG interrupt requests . . . 717
Table 71. RAMCFG register map and reset values . . . 732
Table 72. ICACHE features . . . 736
Table 73. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 738
Table 74. TAG memory dimensioning parameters for direct-mapped cache mode . . . 739
Table 75. ICACHE cacheability for AHB transaction . . . 741
Table 76. Memory configurations . . . 741
Table 77. ICACHE remap region size, base address, and remap address . . . 742
Table 78. ICACHE interrupts . . . 746
Table 79. ICACHE register map and reset values . . . 754
Table 80. DCACHE features . . . 757
Table 81. TAG memory dimensioning parameters . . . 760
Table 82. DCACHE cacheability for AHB transaction . . . 762
Table 83. DCACHE interrupts . . . 767
Table 84. DCACHE register map and reset values . . . 775
Table 85. PWR pin overview . . . 779
Table 86. V_{DDCPU}/V_{DDCORE} voltage levels (for reference only) . . . 782
Table 87. VSW domain function when VDD $_{and}$ VDDA18AON are presents . . . 793
Table 88. CPU1 low-power mode summary . . . 804
Table 89. CPU2 low-power mode summary . . . 805
Table 90. System low-power modes . . . 806
Table 91. Functionalities depending on system operating mode . . . 807
Table 92. External regulators control . . . 810
Table 93. PDD low-power mode control . . . 811
Table 94. Low-power mode exit flags . . . 813
Table 95. CPU1 CSleep mode . . . 817
Table 96. CPU2 CSleep mode . . . 817
Table 97. CPU1 CStop mode . . . 818

Table 98. CPU2 CStop mode . . . 818
Table 99. D1 DStop1 mode . . . 819
Table 100. D1 DStandby mode . . . 820
Table 101. Stop1 mode . . . 821
Table 102. Stop2 mode . . . 822
Table 103. LP-Stop1 mode . . . 822
Table 104. LPLV-Stop1 mode . . . 823
Table 105. LP-Stop2 mode . . . 823
Table 106. LPLV-Stop2 mode . . . 824
Table 107. CPU1_HW_BEN and CPU2_HW_BEN management . . . 825
Table 108. Standby mode . . . 825
Table 109. PWR status flags . . . 828
Table 110. PWR register allocation to RIF resource and access conditions . . . 831
Table 111. PWR register map and reset values . . . 861
Table 112. Pin overview . . . 867
Table 113. IWDG timeout event combinations when TDCID = CPU1 . . . 871
Table 114. IWDG timeout event combinations when TDCID = CPU2 . . . 871
Table 115. Reset coverage summary . . . 874
Table 116. RCC registers with specific reset and specific voltages . . . 876
Table 117. RCC HWRSTSCLRR, RCC_C1HWRSTSCLRR, RCC_C2HWRSTSCLRR flags . . . 878
Table 118. First CPUx boot selection . . . 881
Table 119. HOLD_BOOT state control after system Standby exit . . . 883
Table 120. Oscillator states versus system modes . . . 888
Table 121. Root clock mapping . . . 907
Table 122. APB root clock mapping . . . 909
Table 123. HSI, MSI and HSE states versus system modes . . . 912
Table 124. Resources blocked during clock restore sequence . . . 917
Table 125. Kernel clock distribution overview . . . 919
Table 126. Ratio between clock timers and ck_icn_apbx . . . 922
Table 127. ADC kernel clock source selection . . . 923
Table 128. Gating control of the APB clocks with ck_icn_p_ddrc and ck_icn_p_ddrphyc . . . 926
Table 129. Gating control of ck_icn_s_ddr clock . . . 927
Table 130. DDRCKMOD control description . . . 927
Table 131. DDRCTRL low-power states description . . . 928
Table 132. Recommended configurations when DDR is not shared (DDRSHR = 0) . . . 933
Table 133. Recommended configurations when DDR is shared (DDRSHR = 1) . . . 934
Table 134. MCDCG control bit when DDR is not shared (DDRSHR = 0) . . . 934
Table 135. MCDCG control bit when DDR is shared (DDRSHR = 1) . . . 935
Table 136. DPHYCG control bit when DDR is not shared (DDRSHR = 0) . . . 935
Table 137. DPHYCG control bit when DDR is shared (DDRSHR = 1) . . . 936
Table 138. Shareable peripherals list . . . 939
Table 139. Peripheral bus clock enabling logic . . . 941
Table 140. Peripheral kernel clock enabling logic . . . 942
Table 141. CPU1 mode details . . . 944
Table 142. Bus clock enable details for ck_icn_ahbx with x = 2,5,6 and ck_icn_apby with y = 1,2,3,4,5,dbg . . . 946
Table 143. FMC interface clocks from RCC . . . 947
Table 144. OCTOSPI clocks from RCC . . . 948
Table 145. OCTOSPI interface clocks from DLYBOS1 . . . 948
Table 146. SDMMC clocks from RCC . . . 950
Table 147. SDMMC interface clocks from DLYBSD1/2/3 . . . 951
Table 148. Ethernet clocks from RCC . . . 953

Table 149. Ethernet clocks to/from I/Os . . . 953
Table 150. Active I/O per mode (MII, RMII, RGMII) . . . 953
Table 151. Ethernet MAC speed encoding . . . 955
Table 152. Cortex-A35 interface clocks from RCC . . . 956
Table 153. Debug and Trace subsystem external clocks . . . 957
Table 154. Camera subsystem external clocks . . . 958
Table 155. Camera subsystem internal clocks . . . 959
Table 156. Display subsystem external clocks . . . 960
Table 157. USBH port clocks . . . 961
Table 158. USB2 OTG high speed clocks from RCC and oscillator . . . 962
Table 159. Security clocks from RCC . . . 965
Table 160. Interrupt sources and control . . . 967
Table 161. RCC register allocation to RIF resource and access conditions . . . 973
Table 162. RCC register map and reset values . . . 1151
Table 163. IPCC interface signals . . . 1172
Table 164. Bits used for the communication . . . 1172
Table 165. IPCC security modes . . . 1178
Table 166. Access to channels . . . 1179
Table 167. IPCC register map and reset values . . . 1193
Table 168. Port x bit configurations . . . 1196
Table 169. Secure AF between peripherals and allocated I/Os . . . 1203
Table 170. GPIO protected bits . . . 1204
Table 171. GPIO register map and reset values . . . 1229
Table 172. SYSCFG register map and reset values . . . 1282
Table 173. Peripherals interconnect matrix . . . 1293
Table 174. Peripherals interconnect matrix details . . . 1296
Table 175. Implementation of HPDMA1/2/3 channels . . . 1319
Table 176. HPDMA1/2/3 wake-up in low-power modes . . . 1319
Table 177. Programmed HPDMA1/2/3 request . . . 1320
Table 178. Programmed HPDMA1/2/3 request as a block request . . . 1324
Table 179. HPDMA1/2/3 channel with peripheral early termination . . . 1324
Table 180. Programmed HPDMA1/2/3 request with peripheral early termination . . . 1324
Table 181. Programmed HPDMA1/2/3 trigger . . . 1325
Table 182. Programmed HPDMA source/destination burst . . . 1350
Table 183. Programmed data handling . . . 1355
Table 184. Effect of low-power modes on HPDMA . . . 1379
Table 185. HPDMA interrupt requests . . . 1379
Table 186. HPDMA register map and reset values . . . 1434
Table 187. STM32MP21xx interrupt mapping for Cortex-M33 . . . 1438
Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 . . . 1447
Table 189. GICD register map and reset values . . . 1476
Table 190. GICC register map and reset values . . . 1487
Table 191. GICH register map and reset values . . . 1493
Table 192. GICV register map and reset values . . . 1501
Table 193. EXTI implementation . . . 1504
Table 194. EXTI signals . . . 1505
Table 195. EVG signals . . . 1506
Table 196. EXTI1 wake-up interrupts . . . 1507
Table 197. EXTI2 wake-up interrupts . . . 1510
Table 198. EXTI event input configurations and register control . . . 1512
Table 199. Masking functionality . . . 1515
Table 200. Register protection overview . . . 1516

Table 201. Register protection hierarchy overview . . . 1518
Table 202. EXTIx register map sections . . . 1518
Table 203. EXTI1 register map and reset values . . . 1590
Table 204. EXTI2 register map and reset values . . . 1595
Table 205. CRC internal input/output signals . . . 1600
Table 206. CRC register map and reset values . . . 1605
Table 207. FMC interface interrupts . . . 1608
Table 208. RX and TX delays in lock mode . . . 1612
Table 209. Number of delay elements enabled in each delay unit . . . 1613
Table 210. Octo-SPI interface interrupts . . . 1613
Table 211. OCTOSPI1 CS mapping . . . 1613
Table 212. DLYBOS1 SYSCFG bits . . . 1615
Table 213. OCTOSPI modes and recommended clock configuration . . . 1616
Table 214. SD/eMMC interface interrupts . . . 1622
Table 215. DLYBSD1 SYSCFG bits . . . 1622
Table 216. DLYBSD2 SYSCFG bits . . . 1623
Table 217. DLYBSD3 SYSCFG bits . . . 1624
Table 218. SD speed modes and recommended clock configuration . . . 1624
Table 219. eMMC Speed modes and recommended clock configuration . . . 1625
Table 220. FMC features . . . 1628
Table 221. FMC internal signals . . . 1629
Table 222. FMC memory regions (default mapping) . . . 1631
Table 223. NOR/PSRAM subregion selection . . . 1631
Table 224. NOR/PSRAM External memory address . . . 1632
Table 225. NAND access memory map . . . 1632
Table 226. FMC RIF register properties and definitions . . . 1634
Table 227. Programmable NOR/PSRAM access parameters . . . 1643
Table 228. Non-multiplexed I/O NOR flash memory . . . 1643
Table 229. 16-bit multiplexed I/O NOR flash memory . . . 1644
Table 230. Non-multiplexed I/Os PSRAM/SRAM . . . 1644
Table 231. 16-bit multiplexed I/O PSRAM . . . 1644
Table 232. NOR flash/PSRAM: Example of supported memories and transactions . . . 1645
Table 233. FMC_BCRx bitfields (mode 1) . . . 1649
Table 234. FMC_BTRx bitfields (mode 1) . . . 1649
Table 235. FMC_BCRx bitfields (mode A) . . . 1651
Table 236. FMC_BTRx bitfields (mode A) . . . 1652
Table 237. FMC_BWTRx bitfields (mode A) . . . 1652
Table 238. FMC_BCRx bitfields (mode 2/B) . . . 1654
Table 239. FMC_BTRx bitfields (mode 2/B) . . . 1655
Table 240. FMC_BWTRx bitfields (mode 2/B) . . . 1655
Table 241. FMC_BCRx bitfields (mode C) . . . 1657
Table 242. FMC_BTRx bitfields (mode C) . . . 1657
Table 243. FMC_BWTRx bitfields (mode C) . . . 1658
Table 244. FMC_BCRx bitfields (mode D) . . . 1659
Table 245. FMC_BTRx bitfields (mode D) . . . 1660
Table 246. FMC_BWTRx bitfields (mode D) . . . 1660
Table 247. FMC_BCRx bitfields (muxed mode) . . . 1662
Table 248. FMC_BTRx bitfields (muxed mode) . . . 1663
Table 249. FMC_BCRx bitfields (Synchronous multiplexed read mode) . . . 1668
Table 250. FMC_BTRx bitfields (Synchronous multiplexed read mode) . . . 1669
Table 251. FMC_BCRx bitfields (Synchronous multiplexed write mode) . . . 1670
Table 252. FMC_BTRx bitfields (Synchronous multiplexed write mode) . . . 1671

Table 253. Programmable NAND flash access parameters . . . 1684
Table 254. 8-bit NAND flash memory . . . 1685
Table 255. 16-bit NAND flash memory . . . 1685
Table 256. Supported memories and transactions . . . 1686
Table 257. Number of ECC parity bytes per sector . . . 1691
Table 258. FMC NAND controller Interrupt request . . . 1699
Table 259. HPR relevant bits . . . 1705
Table 260. ECC result relevant bits . . . 1706
Table 261. FMC register map . . . 1724
Table 262. OCTOSPI implementation . . . 1730
Table 263. OCTOSPI input/output pins . . . 1732
Table 264. OCTOSPI internal signals . . . 1732
Table 265. Command/address phase description . . . 1741
Table 266. OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM) . . . 1752
Table 267. Address alignment cases . . . 1759
Table 268. OCTOSPI interrupt requests . . . 1760
Table 269. OCTOSPI register map and reset values . . . 1786
Table 270. SDMMC features . . . 1790
Table 271. SDMMC operation modes SD and SDIO . . . 1793
Table 272. SDMMC operation modes e•MMC . . . 1793
Table 273. SDMMC internal input/output signals . . . 1794
Table 274. SDMMC pins . . . 1795
Table 275. SDMMC Command and data phase selection . . . 1796
Table 276. Command token format . . . 1802
Table 277. Short response with CRC token format . . . 1803
Table 278. Short response without CRC token format . . . 1803
Table 279. Long response with CRC token format . . . 1803
Table 280. Specific Commands overview . . . 1804
Table 281. Command path status flags . . . 1805
Table 282. Command path error handling . . . 1805
Table 283. Data token format . . . 1813
Table 284. Data path status flags and clear bits . . . 1813
Table 285. Data path error handling . . . 1815
Table 286. Data FIFO access . . . 1816
Table 287. Transmit FIFO status flags . . . 1817
Table 288. Receive FIFO status flags . . . 1819
Table 289. AHB and SDMMC_CK clock frequency relation . . . 1823
Table 290. SDIO special operation control . . . 1824
Table 291. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 1828
Table 292. CMD12 use cases . . . 1833
Table 293. Data block gap hardware flow control behavior . . . 1845
Table 294. Hardware flow control selection . . . 1845
Table 295. SDMMC interrupts . . . 1849
Table 296. Response type and SDMMC_RESPxR registers . . . 1856
Table 297. SDMMC register map . . . 1874
Table 298. STM32MP21xx features . . . 1877
Table 299. DLYB functional signals . . . 1879
Table 300. SDMMC DLYB interconnections . . . 1880
Table 301. OCTOSPI DLYB interconnections . . . 1880
Table 302. ADC features . . . 1885
Table 303. ADC input/output pins . . . 1887

Table 304. ADC internal input/output signals . . . 1887
Table 305. ADC1 interconnection . . . 1887
Table 306. ADC2 interconnection . . . 1889
Table 307. Configuring the trigger polarity for regular external triggers . . . 1910
Table 308. Configuring the trigger polarity for injected external triggers . . . 1910
Table 309. TSAR timings depending on resolution . . . 1912
Table 310. Offset computation versus data resolution . . . 1916
Table 311. 12-bit data formats . . . 1918
Table 312. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0) . . . 1919
Table 313. Analog window watchdog features . . . 1927
Table 314. Analog watchdog 1 channel selection . . . 1928
Table 315. Analog watchdog 1, 2, 3 comparison . . . 1929
Table 316. Maximum output results versus N and M . . . 1934
Table 317. Oversampler operating mode summary . . . 1938
Table 318. DELAY bits versus ADC resolution . . . 1938
Table 319. ADC interrupts . . . 1942
Table 320. ADC register map and reset values . . . 1978
Table 321. ADC register map and reset values (master and slave ADC common registers) . . . 1981
Table 322. DTS internal input/output signals . . . 1983
Table 323. SDA slave registers . . . 1988
Table 324. SDA TS control register (SDATS_CR) . . . 1988
Table 325. SDA TS configuration register (SDATS_CFGR) . . . 1989
Table 326. SDA TS data register (SDATS_DR) . . . 1990
Table 327. SDA TS timer register (SDATS_TIMERR) . . . 1991
Table 328. Output resolution configuration . . . 1996
Table 329. DTS interrupts . . . 1997
Table 330. DTS register memory map . . . 1998
Table 331. PVT common register map and reset values . . . 2018
Table 332. PVT IRQ register map and reset values . . . 2018
Table 333. DTS TSC clock synthesizer register map and reset values . . . 2019
Table 334. DTS TS individual register map and reset values . . . 2021
Table 335. VREFBUF typical values . . . 2023
Table 336. VREF buffer modes . . . 2024
Table 337. VREFBUF register map and reset values . . . 2027
Table 338. MDF features . . . 2029
Table 339. MDF external pins . . . 2032
Table 340. MDF internal signals . . . 2032
Table 341. MDF trigger connections . . . 2032
Table 342. MDF break connections . . . 2033
Table 343. MDF ADC data connections . . . 2033
Table 344. Control of the common clock generation . . . 2041
Table 345. Clock constraints with respect to the incoming stream . . . 2042
Table 346. Data size according to CIC order and CIC decimation values . . . 2049
Table 347. Maximum decimation ratio versus order and input data size . . . 2050
Table 348. Possible gain values . . . 2051
Table 349. Recommended maximum gain values versus CIC decimation ratios . . . 2053
Table 350. Most common microphone settings . . . 2053
Table 351. HPF 3 dB cut-off frequencies examples . . . 2055
Table 352. Register protection summary . . . 2076
Table 353. Effect of low-power modes on MDF . . . 2077
Table 354. MDF interrupt requests . . . 2079

Table 355. Examples of MDF settings for microphone capture . . . 2080
Table 356. Programming sequence . . . 2080
Table 357. Output signal levels . . . 2087
Table 358. MDF register map and reset values . . . 2113
Table 359. Camera subsystem RIF peripheral ID . . . 2121
Table 360. DCMI input/output pins . . . 2127
Table 361. DCMI internal input/output signals . . . 2127
Table 362. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 2129
Table 363. Positioning of captured data bytes in 32-bit words (10-bit width) . . . 2129
Table 364. Positioning of captured data bytes in 32-bit words (12-bit width) . . . 2129
Table 365. Positioning of captured data bytes in 32-bit words (14-bit width) . . . 2130
Table 366. Data storage in monochrome progressive video format . . . 2135
Table 367. Data storage in RGB progressive video format . . . 2136
Table 368. Data storage in YCbCr progressive video format . . . 2136
Table 369. Data storage in YCbCr progressive video format - Y extraction mode . . . 2136
Table 370. DCMI interrupts . . . 2137
Table 371. DCMI register map and reset values . . . 2148
Table 372. Available pipelines . . . 2150
Table 373. Glossary . . . 2151
Table 374. DCMIPP input/output pads . . . 2156
Table 375. DCMIPP input/output pins . . . 2156
Table 376. DCMIPP clocks . . . 2156
Table 377. DCMIPP resets . . . 2157
Table 378. Parallel interface maximum resolution (80 MHz) . . . 2158
Table 379. Supported pixel formats . . . 2160
Table 380. DCMIPP_CMCR bit function . . . 2162
Table 381. DCMIPP_PRCR bit function . . . 2165
Table 382. DCMIPP_PRESCR and DCMIPP_PRESUR bit function . . . 2165
Table 383. DCMIPP_CMTPGCR1 and DCMIPP_CMTPGCR2 bit function . . . 2167
Table 384. DCMIPP_PxFCTCR bit function . . . 2175
Table 385. DCMIPP_P0PPCR bit function . . . 2178
Table 386. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function . . . 2181
Table 387. DCMIPP_P1SRCR bit function . . . 2183
Table 388. DCMIPP_P1BPRCR and DCMIPP_P1BPRSR bit function . . . 2184
Table 389. DCMIPP_P1DECR bit function . . . 2185
Table 390. DCMIPP_P1BLCCR bit function . . . 2186
Table 391. DCMIPP_P1EXCR1 and DCMIPP_P1EXCR2 bit function . . . 2187
Table 392. DCMIPP_P1DMCR bit function . . . 2189
Table 393. DCMIPP_PxCCyy (yy = R0, R1, G0, G1, B0, B1) bit function . . . 2190
Table 394. Color conversion: examples of coefficients . . . 2191
Table 395. DCMIPP_P1CTCR1,2,3 bit function . . . 2193
Table 396. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function . . . 2194
Table 397. Statistics extraction: collected data vs. modes . . . 2196
Table 398. DCMIPP_P1HSCR, DCMIPP_P1HSSR, DCMIPP_P1HSMAR1, DCMIPP_P1HSMAR2, and DCMIPP_P1HSSMAR bit function . . . 2198
Table 399. DCMIPP_PxCRSTR and DCMIPP_PxCRSZR bit function . . . 2205
Table 400. DCMIPP_PxDCCR bit function . . . 2206
Table 401. DCMIPP_PxDSCR, DCMIPP_PxDSRTIOR, DCMIPP_PxDSSZR . . . 2207
Table 402. DCMIPP_PxPPCR bit function . . . 2209
Table 403. Parallel interface input pixel formats . . . 2222
Table 404. Correspondence between index and DCMIPP_PRCR register values . . . 2223

Table 405. Parallel interface input pixel formats . . . 2224
Table 406. Pixel pipe output pixel format . . . 2224
Table 407. Dump pipe OUTPUT pixel formats . . . 2226
Table 408. Shadow and physical registers . . . 2233
Table 409. DCMIPP low power modes . . . 2235
Table 410. DCMIPP interrupts . . . 2238
Table 411. Event connection . . . 2239
Table 412. DCMIPP registers organization . . . 2241
Table 413. DCMIPP register map and reset values . . . 2343
Table 414. Glossary of terms . . . 2357
Table 415. Effect of low-power modes on CSI-2 Host controller . . . 2376
Table 416. List of errors detected by the CSI-2 Host controller . . . 2376
Table 417. List of events that trigger an interrupt . . . 2377
Table 418. CSI-2 Host and PHY register map and reset values . . . 2415
Table 419. PSSI input/output pins . . . 2421
Table 420. PSSI internal input/output signals . . . 2421
Table 421. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 2422
Table 422. Positioning of captured data bytes in 32-bit words (16-bit width) . . . 2423
Table 423. PSSI interrupt requests . . . 2426
Table 424. PSSI register map and reset values . . . 2433
Table 425. Display subsystem RIF peripheral ID . . . 2440
Table 426. LTDC pins . . . 2445
Table 427. Clock domain for each register . . . 2446
Table 428. LTDC register access and updated durations . . . 2448
Table 429. Pixel data mapping versus color format . . . 2451
Table 430. LTDC interrupt requests . . . 2468
Table 431. LTDC register map and reset values . . . 2504
Table 432. RNG internal input/output signals . . . 2513
Table 433. RNG interrupt requests . . . 2521
Table 434. RNG configurations . . . 2522
Table 435. Additional health test configurations . . . 2522
Table 436. Configuration selection . . . 2522
Table 437. RNG register map and reset map . . . 2529
Table 438. CRYP1 versus SAES features . . . 2533
Table 439. SAES internal input/output signals . . . 2534
Table 440. SAES approved AES symmetric key functions . . . 2535
Table 441. Counter mode initialization vector definition . . . 2545
Table 442. Initialization of IV registers in GCM mode . . . 2548
Table 443. GCM last block definition . . . 2548
Table 444. Initialization of IV registers in CCM mode . . . 2554
Table 445. SAES data swapping example . . . 2563
Table 446. Key endianness in SAES_KEYRx registers (128/192/256-bit keys) . . . 2565
Table 447. IVI bitfield spread over SAES_IVRx registers . . . 2567
Table 448. SAES interrupt requests . . . 2569
Table 449. Processing latency for ECB, CBC and CTR . . . 2570
Table 450. Processing latency for GCM and CCM . . . 2571
Table 451. SAES register map and reset values . . . 2586
Table 452. CRYP versus SAES features . . . 2590
Table 453. CRYP internal input/output signals . . . 2592
Table 454. CRYP approved AES symmetric key functions . . . 2592
Table 455. CRYP approved DES/TDES symmetric key functions . . . 2592
Table 456. Counter mode initialization vector definition . . . 2606

Table 457. GCM mode IVI registers initialization . . . 2609
Table 458. GCM last block definition . . . 2609
Table 459. CCM mode IVI registers initialization . . . 2616
Table 460. DES/TDES data swapping example . . . 2621
Table 461. AES data swapping example . . . 2622
Table 462. Key endianness in CRYP_KxR/LR registers (AES 128/192/256-bit keys) . . . 2624
Table 463. Key endianness in CRYP_KxR/LR registers (DES K1 and TDES K1/2/3) . . . 2624
Table 464. Initialization vector endianness in CRYP_IVxR registers (AES) . . . 2625
Table 465. Initialization vector endianness in CRYP_IVxR registers (DES/TDES) . . . 2625
Table 466. CRYP interrupt requests . . . 2627
Table 467. Processing latency for ECB, CBC and CTR . . . 2628
Table 468. Processing latency for GCM and CCM (in clock cycles) . . . 2629
Table 469. CRYP register map and reset values . . . 2644
Table 470. HASH internal input/output signals . . . 2649
Table 471. Information on supported hash algorithms . . . 2649
Table 472. Hash processor outputs . . . 2653
Table 473. Processing time (in clock cycle) . . . 2660
Table 474. HASH interrupt requests . . . 2661
Table 475. HASH1 register map and reset values . . . 2671
Table 476. OTFDEC internal input/output signals . . . 2674
Table 477. OTFDEC interrupt requests . . . 2678
Table 478. OTFDEC register map and reset values . . . 2691
Table 479. Internal input/output signals . . . 2696
Table 480. PKA integer arithmetic functions list . . . 2697
Table 481. PKA prime field (Fp) elliptic curve functions list . . . 2698
Table 482. Montgomery parameter computation . . . 2704
Table 483. Modular addition . . . 2705
Table 484. Modular subtraction . . . 2705
Table 485. Montgomery multiplication . . . 2706
Table 486. Modular exponentiation (normal mode) . . . 2707
Table 487. Modular exponentiation (fast mode) . . . 2707
Table 488. Modular exponentiation (protected mode) . . . 2708
Table 489. Modular inversion . . . 2708
Table 490. Modular reduction . . . 2709
Table 491. Arithmetic addition . . . 2709
Table 492. Arithmetic subtraction . . . 2709
Table 493. Arithmetic multiplication . . . 2710
Table 494. Arithmetic comparison . . . 2710
Table 495. CRT exponentiation . . . 2711
Table 496. Point on elliptic curve Fp check . . . 2712
Table 497. ECC Fp scalar multiplication . . . 2712
Table 498. ECDSA sign - Inputs . . . 2714
Table 499. ECDSA sign - Outputs . . . 2714
Table 500. Extended ECDSA sign - Extra outputs . . . 2715
Table 501. ECDSA verification - Inputs . . . 2715
Table 502. ECDSA verification - Outputs . . . 2716
Table 503. ECC complete addition . . . 2716
Table 504. ECC double base ladder . . . 2717
Table 505. ECC projective to affine . . . 2718
Table 506. Family of supported curves for ECC operations . . . 2719
Table 507. Modular exponentiation . . . 2720
Table 508. ECC scalar multiplication . . . 2720

Table 509. ECDSA signature average computation time . . . 2721
Table 510. ECDSA verification average computation times . . . 2721
Table 511. ECC double base ladder average computation times . . . 2721
Table 512. ECC projective to affine average computation times . . . 2721
Table 513. ECC complete addition average computation times . . . 2721
Table 514. Point on elliptic curve Fp check average computation times . . . 2721
Table 515. Montgomery parameters average computation times . . . 2722
Table 516. PKA interrupt requests . . . 2722
Table 517. PKA register map and reset values . . . 2729
Table 518. TIM input/output pins . . . 2732
Table 519. TIM internal input/output signals . . . 2732
Table 520. Interconnect to the tim_ti1 input multiplexer . . . 2733
Table 521. Interconnect to the tim_ti2 input multiplexer . . . 2733
Table 522. Interconnect to the tim_ti3 input multiplexer . . . 2734
Table 523. Interconnect to the tim_ti4 input multiplexer . . . 2734
Table 524. Internal trigger connection . . . 2734
Table 525. Interconnect to the tim_etr input multiplexer . . . 2735
Table 526. Timer break interconnect . . . 2735
Table 527. Timer break2 interconnect . . . 2736
Table 528. System break interconnect . . . 2736
Table 529. CCR and ARR register change dithering pattern . . . 2768
Table 530. CCR register change dithering pattern in center-aligned PWM mode . . . 2769
Table 531. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 2781
Table 532. Break protection disarming conditions . . . 2783
Table 533. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 2791
Table 534. Counting direction versus encoder signals and polarity settings . . . 2795
Table 535. DMA request . . . 2816
Table 536. Effect of low-power modes on TIM1/TIM8 . . . 2817
Table 537. Interrupt requests . . . 2817
Table 538. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 2844
Table 539. TIMx register map and reset values . . . 2871
Table 540. STM32MP21xx general purpose timers . . . 2875
Table 541. TIM input/output pins . . . 2877
Table 542. TIM internal input/output signals . . . 2877
Table 543. Interconnect to the tim_ti1 input multiplexer . . . 2878
Table 544. Interconnect to the tim_ti2 input multiplexer . . . 2878
Table 545. Interconnect to the tim_ti3 input multiplexer . . . 2878
Table 546. Interconnect to the tim_ti4 input multiplexer . . . 2879
Table 547. TIMx internal trigger connection . . . 2879
Table 548. Interconnect to the tim_etr input multiplexer . . . 2879
Table 549. CCR and ARR register change dithering pattern . . . 2911
Table 550. CCR register change dithering pattern in center-aligned PWM mode . . . 2912
Table 551. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 2921
Table 552. Counting direction versus encoder signals and polarity settings . . . 2926
Table 553. DMA request . . . 2950
Table 554. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 2950
Table 555. Interrupt requests . . . 2951
Table 556. Output control bit for standard tim_ocx channels . . . 2972
Table 557. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 2987
Table 558. TIM internal input/output signals . . . 2991
Table 559. TIMx_ARR register change dithering pattern . . . 3001

Table 560. DMA request . . . 3002
Table 561. Effect of low-power modes on TIM6/TIM7 . . . 3002
Table 562. Interrupt request . . . 3002
Table 563. TIMx register map and reset values . . . 3012
Table 564. TIM input/output pins . . . 3017
Table 565. TIM internal input/output signals . . . 3017
Table 566. Interconnect to the tim_ti1 input multiplexer . . . 3018
Table 567. Interconnect to the tim_ti2 input multiplexer . . . 3018
Table 568. TIMx internal trigger connection . . . 3018
Table 569. CCR and ARR register change dithering pattern . . . 3036
Table 570. Effect of low-power modes on TIM10/TIM11/TIM12/TIM13/TIM14 . . . 3045
Table 571. Interrupt requests . . . 3045
Table 572. Output control bit for standard tim_ocx channels . . . 3058
Table 573. TIM12 register map and reset values . . . 3066
Table 574. Output control bit for standard tim_ocx channels . . . 3075
Table 575. TIM10/TIM11/TIM13/TIM14 register map and reset values . . . 3082
Table 576. TIM input/output pins . . . 3087
Table 577. TIM internal input/output signals . . . 3088
Table 578. Interconnect to the tim_ti1 input multiplexer . . . 3088
Table 579. Interconnect to the tim_ti2 input multiplexer . . . 3089
Table 580. TIMx internal trigger connection . . . 3089
Table 581. Timer break interconnect . . . 3090
Table 582. System break interconnect . . . 3090
Table 583. CCR and ARR register change dithering pattern . . . 3110
Table 584. Break protection disarming conditions . . . 3119
Table 585. DMA request . . . 3130
Table 586. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 3131
Table 587. Interrupt requests . . . 3131
Table 588. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 3147
Table 589. TIM15 register map and reset values . . . 3163
Table 590. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 3177
Table 591. TIM16/TIM17 register map and reset values . . . 3194
Table 592. STM32MP21xx LPTIM features . . . 3198
Table 593. LPTIM1/2/3/4 input/output pins . . . 3200
Table 594. LPTIM5 input/output pins . . . 3200
Table 595. LPTIM1/2/3/4 internal signals . . . 3201
Table 596. LPTIM5 internal signals . . . 3201
Table 597. LPTIM1/2/3/4/5 external trigger connections . . . 3202
Table 598. LPTIM1/2/3/4/5 input 1 connections . . . 3202
Table 599. LPTIM1/2 input 2 connections . . . 3202
Table 600. LPTIM1/2/3/4 input capture 1 connections . . . 3202
Table 601. LPTIM1/2/3 input capture 2 connections . . . 3203
Table 602. Prescaler division ratios . . . 3204
Table 603. Encoder counting scenarios . . . 3211
Table 604. Input capture Glitch filter latency (in counter step unit) . . . 3215
Table 605. Interrupt events . . . 3221
Table 606. LPTIM register map and reset values . . . 3248
Table 607. STGENC register map and reset values . . . 3266
Table 608. STGENR register map and reset values . . . 3267
Table 609. STM32MP21xx IWDG features . . . 3271

Table 610. IWDG delays versus actions . . . 3272
Table 611. IWDG internal input/output signals . . . 3273
Table 612. Effect of low power modes on IWDG . . . 3278
Table 613. IWDG interrupt request . . . 3280
Table 614. IWDG register map and reset values . . . 3288
Table 615. STM32MP21xx WWDG features . . . 3289
Table 616. WWDG internal input/output signals . . . 3290
Table 617. WWDG register map and reset values . . . 3296
Table 618. RTC input/output pins . . . 3301
Table 619. RTC internal input/output signals . . . 3301
Table 620. RTC interconnection . . . 3302
Table 621. RTC pin PC13 configuration . . . 3302
Table 622. PI8 configuration . . . 3305
Table 623. RTC_OUT mapping . . . 3306
Table 624. RTC resource number allocation . . . 3309
Table 625. Effect of low-power modes on RTC . . . 3323
Table 626. RTC pins functionality over modes . . . 3323
Table 627. Nonsecure interrupt requests . . . 3324
Table 628. Secure interrupt requests . . . 3325
Table 629. RTC register map and reset values . . . 3361
Table 630. TAMP input/output pins . . . 3367
Table 631. TAMP internal input/output signals . . . 3367
Table 632. TAMP interconnection . . . 3368
Table 633. CID regions . . . 3372
Table 634. RIF TAMP resource number allocation . . . 3373
Table 635. Device resource x tamper protection . . . 3377
Table 636. Active tamper output change period . . . 3381
Table 637. Minimum ATPER value . . . 3382
Table 638. Active tamper filtered pulse duration . . . 3383
Table 639. Effect of low-power modes on TAMP . . . 3385
Table 640. TAMP pins functionality over modes . . . 3385
Table 641. Nonsecure interrupt requests . . . 3385
Table 642. Secure interrupt requests . . . 3386
Table 643. TAMP register map and reset values . . . 3423
Table 644. I2C implementation . . . 3427
Table 645. I2C input/output pins . . . 3428
Table 646. I2C internal input/output signals . . . 3429
Table 647. I2C1 interconnection . . . 3429
Table 648. I2C2 interconnection . . . 3429
Table 649. I2C3 interconnection . . . 3430
Table 650. Comparison of analog and digital filters . . . 3433
Table 651. I²C-bus and SMBus specification data setup and hold times . . . 3435
Table 652. I2C configuration . . . 3439
Table 653. I²C-bus and SMBus specification clock timings . . . 3450
Table 654. Timing settings for f_{I2CCLK} of 8 MHz . . . 3460
Table 655. Timing settings for f_{I2CCLK} of 16 MHz . . . 3460
Table 656. SMBus timeout specifications . . . 3462
Table 657. SMBus with PEC configuration . . . 3464
Table 658. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 3465
Table 659. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 3465
Table 660. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 3465
Table 661. Effect of low-power modes to I2C . . . 3475

Table 662. I2C interrupt requests . . . 3476
Table 663. I2C register map and reset values . . . 3494
Table 664. I3C instantiation . . . 3498
Table 665. I3C wake-up . . . 3498
Table 666. I3C autonomous mode with DMA in Stop mode(s) . . . 3498
Table 667. I3C FIFOs implementation . . . 3499
Table 668. I3C interrupt(s) . . . 3499
Table 669. I3C peripheral controller/target features versus MIPI v1.1 . . . 3500
Table 670. I3C input/output pins . . . 3501
Table 671. I3C internal input/output signals . . . 3501
Table 672. I3C register usage . . . 3514
Table 673. I3C registers/fields usage versus controller state . . . 3515
Table 674. I3C registers/fields usage versus target state . . . 3518
Table 675. List of supported I3C CCCs, as controller/target . . . 3521
Table 676. I3C controller error management . . . 3553
Table 677. I3C target error management . . . 3555
Table 678. Effect of low-power modes . . . 3559
Table 679. I3C register map and reset values . . . 3606
Table 680. Instance implementation on STM32MP21xx . . . 3611
Table 681. USART/LPUART features . . . 3611
Table 682. USART/UART input/output pins . . . 3614
Table 683. USART internal input/output signals . . . 3615
Table 684. USART interconnection (USART1/2/3/6 and UART4/5/7) . . . 3615
Table 685. Noise detection from sampled data . . . 3627
Table 686. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 3631
Table 687. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 3631
Table 688. USART frame formats . . . 3636
Table 689. Effect of low-power modes on the USART . . . 3657
Table 690. USART interrupt requests . . . 3658
Table 691. USART register map and reset values . . . 3700
Table 692. Instance implementation on STM32MP21xx . . . 3703
Table 693. USART/LPUART features . . . 3703
Table 694. LPUART input/output pins . . . 3706
Table 695. LPUART internal input/output signals . . . 3706
Table 696. LPUART interconnections (LPUART1) . . . 3707
Table 697. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 3718
Table 698. Tolerance of the LPUART receiver . . . 3719
Table 700. Effect of low-power modes on the LPUART . . . 3729
Table 701. LPUART interrupt requests . . . 3730
Table 702. LPUART register map and reset values . . . 3758
Table 703. SPI features . . . 3761
Table 704. SPI/I2S input/output pins . . . 3763
Table 705. SPI internal input/output signals . . . 3764
Table 706. SPI interconnection (SPI1 to SPI6) . . . 3764
Table 707. Effect of low-power modes on the SPI . . . 3793
Table 708. SPI wake-up and interrupt requests . . . 3794
Table 709. Bitfields usable in PCM/I2S mode . . . 3796
Table 710. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 3805
Table 711. Serial data line swapping . . . 3805
Table 712. CLKGEN programming examples for usual I2S frequencies . . . 3809
Table 713. I2S interrupt requests . . . 3818
Table 714. SPI register map and reset values . . . 3840

Table 715. SAI features . . . 3843
Table 716. SAI internal input/output signals . . . 3845
Table 717. SAI input/output pins . . . 3845
Table 718. External synchronization selection . . . 3848
Table 719. MCLK_x activation conditions . . . 3853
Table 720. Clock generator programming examples . . . 3856
Table 721. SAI_A configuration for TDM mode . . . 3863
Table 722. TDM frame configuration examples . . . 3865
Table 723. SOPD pattern . . . 3869
Table 724. Parity bit calculation . . . 3869
Table 725. Audio sampling frequency versus symbol rates . . . 3870
Table 726. SAI interrupt sources . . . 3879
Table 727. SAI register map and reset values . . . 3910
Table 728. SPDIFRX internal input/output signals . . . 3913
Table 729. SPDIFRX pins . . . 3913
Table 730. Transition sequence for preamble . . . 3919
Table 731. Minimum spdifrx_ker_ck frequency versus audio sampling rate . . . 3929
Table 732. Conditions of spdifrx_symb_ck generation . . . 3930
Table 733. Bit field property versus SPDIFRX state . . . 3933
Table 734. SPDIFRX interface register map and reset values . . . 3946
Table 735. CAN subsystem I/O signals . . . 3948
Table 736. CAN subsystem I/O pins . . . 3949
Table 737. CAN triggers . . . 3949
Table 738. Main features . . . 3951
Table 739. DLC coding in FDCAN . . . 3956
Table 740. Example of filter configuration for Rx buffers . . . 3969
Table 741. Example of filter configuration for Debug messages . . . 3970
Table 742. Possible configurations for frame transmission . . . 3970
Table 743. Tx buffer/FIFO - Queue element size . . . 3971
Table 744. First byte of level 1 reference message . . . 3981
Table 745. First four bytes of level 2 reference message . . . 3982
Table 746. First four bytes of level 0 reference message . . . 3982
Table 747. TUR configuration example . . . 3983
Table 748. System matrix, Node A . . . 3988
Table 749. Trigger list, Node A . . . 3989
Table 750. Number of data bytes transmitted with a reference message . . . 3996
Table 751. Rx buffer and FIFO element . . . 4003
Table 752. Rx buffer and FIFO element description . . . 4003
Table 753. Tx buffer and FIFO element . . . 4005
Table 754. Tx buffer element description . . . 4005
Table 755. Tx Event FIFO element . . . 4007
Table 756. Tx Event FIFO element description . . . 4007
Table 757. Standard message ID filter element . . . 4008
Table 758. Standard message ID filter element field description . . . 4009
Table 759. Extended message ID filter element . . . 4010
Table 760. Extended message ID filter element field description . . . 4010
Table 761. Trigger memory element . . . 4011
Table 762. Trigger memory element description . . . 4011
Table 763. FDCAN register map and reset values . . . 4049
Table 764. FDCAN TT register map and reset values . . . 4071
Table 765. CCU register map and reset values . . . 4077
Table 766. Implementation parameters . . . 4081

Table 767. USB2 host high-speed port pins . . . 4083
Table 768. USB2 host high-speed port internal signals . . . 4083
Table 769. USB2 host high-speed host memory requirements . . . 4085
Table 770. USB2 host high-speed host interrupts . . . 4085
Table 771. USB2 host high-speed host wake-up event . . . 4085
Table 772. connections to SYSCFG block . . . 4087
Table 773. USB2 OTG high-speed port pins . . . 4090
Table 774. USB2 OTG high-speed port internal signals . . . 4091
Table 775. OTG high-speed port clocks . . . 4092
Table 776. USB2 OTG high-speed port memory requirements . . . 4093
Table 777. USB2 OTG high-speed port interrupt . . . 4093
Table 778. USB2 OTG high-speed port wake-up event . . . 4093
Table 779. Connections to SYSCFG block . . . 4096
Table 780. USB debug signals . . . 4097
Table 781. OTG speeds supported . . . 4099
Table 782. OTG implementation . . . 4101
Table 783. OTG input/output pins . . . 4102
Table 784. OTG input/output signals . . . 4103
Table 785. Compatibility of STM32 low power modes with the OTG . . . 4113
Table 786. Core global control and status registers (CSRs) . . . 4121
Table 787. Host-mode control and status registers (CSRs) . . . 4122
Table 788. Device-mode control and status registers . . . 4123
Table 789. Data FIFO (DFIFO) access register map . . . 4125
Table 790. Power and clock gating control and status registers . . . 4126
Table 791. TRDT values . . . 4132
Table 792. Minimum duration for soft disconnect . . . 4175
Table 793. OTG register map and reset values . . . 4201
Table 794. USBH register map and reset values . . . 4277
Table 795. PHY interface selection . . . 4283
Table 796. Ethernet peripheral pins . . . 4291
Table 797. Ethernet internal input/output signals . . . 4292
Table 798. Fixed priority scheme for DMA channels . . . 4307
Table 799. Routing DA/SA failed packets to Rx queue . . . 4312
Table 800. Routing DA/SA pass but VLAN filter fail packets to Rx queue . . . 4312
Table 801. Priority routing of filter pass packets when OMCBCQ = 0 . . . 4312
Table 802. Priority routing of filter pass packets when OMCBCQ = 1 . . . 4313
Table 803. Weight for DMA channels . . . 4315
Table 804. Example of credit value per transmit cycle . . . 4316
Table 805. Double VLAN processing features in Tx path . . . 4319
Table 806. Double VLAN processing in Rx path . . . 4320
Table 807. VLAN insertion or replacement based on VLTI bit . . . 4321
Table 808. Destination address filtering . . . 4325
Table 809. Source address filtering . . . 4326
Table 810. VLAN match status . . . 4327
Table 811. OTS and ITS bit values with at least one perfect filter enabled . . . 4330
Table 812. OTS and ITS bit values with only VLAN Hash filter enabled . . . 4331
Table 813. Rx Queue routing table for Unicast-tagged packets . . . 4332
Table 814. Ordinary clock: PTP messages for snapshot . . . 4335
Table 815. End-to-end transparent clock: PTP messages for snapshot . . . 4336
Table 816. Peer-to-peer transparent clock: PTP messages for snapshot . . . 4337
Table 817. Egress and ingress latency for PHY interfaces . . . 4341
Table 818. Minimum PTP clock frequency example . . . 4341

Table 819. Message format defined in IEEE 1588-2008 . . . 4342
Table 820. Message format defined in IEEE 1588-2008 . . . 4343
Table 821. IPv6-UDP PTP packet fields required for control and status . . . 4344
Table 822. Ethernet PTP packet fields required for control and status . . . 4345
Table 823. Timestamp Snapshot Dependency on ETH_MACTSCR bits . . . 4346
Table 824. PTP message generation criteria . . . 4352
Table 825. Common PTP message header fields . . . 4354
Table 826. MAC Transmit PTP mode and one-step timestamping operation . . . 4357
Table 827. External memory used for holding the two gate-control lists . . . 4362
Table 828. GCL and associated registers - BTR and CTR . . . 4366
Table 829. GCL and associated registers - BTR and CTR, execution time > CycleTime . . . 4367
Table 830. GCL and Associated Registers - BTR and CTR . . . 4369
Table 831. Extending to New List By Truncating the Current List . . . 4370
Table 832. Switching to New List By Extending the Current List . . . 4371
Table 833. Gate Control List When FPE is disabled . . . 4373
Table 834. Gate Control List When FPE is enabled . . . 4373
Table 835. Valid SMD Values of mPacket . . . 4376
Table 836. Valid frag_count values . . . 4376
Table 837. Current and Previous SMD Values . . . 4377
Table 838. Current and previous SMD values . . . 4377
Table 839. Preemption Control Values on MRI Interface for Various Frame Types . . . 4381
Table 840. MMC Counters and Associated Interrupt Registers . . . 4382
Table 841. Transmit checksum offload engine functions for different packet types . . . 4391
Table 842. Receive checksum offload engine functions for different packet types . . . 4394
Table 843. TSO: TCP and IP header fields . . . 4398
Table 844. Pause packet fields . . . 4403
Table 845. Tx MAC flow control . . . 4404
Table 846. Rx MAC flow control . . . 4405
Table 847. Size of the maximum receive packet . . . 4409
Table 848. MCD clock selection . . . 4412
Table 849. MDIO Clause 45 frame structure . . . 4413
Table 850. MDIO Clause 22 frame structure . . . 4414
Table 851. Remote wake-up packet filter register . . . 4427
Table 852. Description of the remote wake-up filter fields . . . 4427
Table 853. Remote wake-up packet and PMT interrupt generation . . . 4429
Table 854. Transfer complete interrupt behavior . . . 4438
Table 855. TDES0 normal descriptor (read format) . . . 4469
Table 856. TDES1 normal descriptor (read format) . . . 4470
Table 857. TDES2 normal descriptor (read format) . . . 4470
Table 858. TDES3 normal descriptor (read format) . . . 4471
Table 859. TDES0 normal descriptor (write-back format) . . . 4474
Table 860. TDES1 normal descriptor (write-back format) . . . 4474
Table 861. TDES2 normal descriptor (write-back format) . . . 4475
Table 862. TDES3 normal descriptor (write-back format) . . . 4475
Table 863. TDES0 context descriptor . . . 4479
Table 864. TDES1 context descriptor . . . 4479
Table 865. TDES2 context descriptor . . . 4480
Table 866. TDES3 context descriptor . . . 4480
Table 867. RDES0 normal descriptor (read format) . . . 4483
Table 868. RDES1 normal descriptor (read format) . . . 4483
Table 869. RDES2 normal descriptor (read format) . . . 4483
Table 870. RDES3 normal descriptor (read format) . . . 4484

Table 871. RDES0 normal descriptor (write-back format) . . . 4485
Table 872. RDES1 normal descriptor (write-back format) . . . 4486
Table 873. RDES2 normal descriptor (write-back format) . . . 4488
Table 874. RDES3 normal descriptor (write-back format) . . . 4490
Table 875. RDES0 context descriptor . . . 4493
Table 876. RDES1 context descriptor . . . 4494
Table 877. RDES2 context descriptor . . . 4494
Table 878. RDES3 context descriptor . . . 4494
Table 879. Enhanced normal descriptor (read format) . . . 4495
Table 880. ETH_DMA common register map and reset values . . . 4532
Table 881. ETH_DMA_CH register map and reset values . . . 4533
Table 882. ETH_DMA_CH1 register map and reset values . . . 4535
Table 883. ETH_DMA_CH2 register map and reset values . . . 4536
Table 884. ETH_DMA_CH3 register map and reset values . . . 4538
Table 885. ETH_MTL register map and reset values . . . 4568
Table 886. Giant Packet Status based on S2KP and JE Bits . . . 4578
Table 887. Packet Length based on the CST and ACS bits . . . 4578
Table 888. Ethernet MAC register map and reset values . . . 4721
Table 889. HDP signal muxing . . . 4738
Table 890. HDP register map and reset values . . . 4742
Table 891. Key internal signals . . . 4746
Table 892. SERF resources mapping . . . 4751
Table 893. SERC register map and reset values . . . 4763
Table 894. JTAG/Serial-wire debug port pins . . . 4767
Table 895. Trace port pins . . . 4768
Table 896. Serial-wire trace port pins . . . 4768
Table 897. Trigger pins . . . 4768
Table 898. Authentication signal connectivity . . . 4772
Table 899. Authentication signal initial states . . . 4772
Table 900. CLTAPC registers . . . 4773
Table 901. AP0 (DBGMCU APB) authentication behavior . . . 4776
Table 902. AP1 (debug APB) authentication behavior . . . 4776
Table 903. AP2-3 authentication behavior . . . 4776
Table 904. Debug subsystem ROM table . . . 4777
Table 905. Cortex-A35 subsystem ROM table . . . 4778
Table 906. DBGMCU ROM table . . . 4778
Table 907. Trace subsystem CTI_0 inputs . . . 4783
Table 908. Trace subsystem CTI_0 outputs . . . 4783
Table 909. Trace subsystem CTI_1 inputs . . . 4784
Table 910. Trace subsystem CTI_1 outputs . . . 4784
Table 911. Cortex-A35_0 CTI inputs . . . 4784
Table 912. Cortex-A35_0 CTI outputs . . . 4785
Table 913. Cortex-M33 CTI inputs . . . 4785
Table 914. Cortex-M33 CTI outputs . . . 4785
Table 915. Replicator allocation . . . 4790
Table 916. STM extended stimulus port memory map . . . 4792
Table 917. STM trace packet ID mapping to AXI masters . . . 4792
Table 918. Hardware event mapping 0 to 3 . . . 4793
Table 919. Hardware event mapping 4 to 7 . . . 4795
Table 920. Hardware event mapping 8 to 9 . . . 4797
Table 921. IWDG1 behavior in debug . . . 4801
Table 922. DBGMCU register map and reset values . . . 4817

Table 923. Cortex-M33 CPU ROM table 4822
Table 924. Document revision history 4833

List of figures

Figure 1. STM32MP21xx interconnect top view . . . 171
Figure 2. STNoC-based MPU-side AXI interconnect . . . 176
Figure 3. STNoC internal router distribution (request path) and frequency distribution . . . 177
Figure 4. MLAHB interconnect for STM32MP21xx . . . 180
Figure 5. Memory map . . . 185
Figure 6. Cortex-M33 sub-system . . . 229
Figure 7. RIF principle . . . 237
Figure 8. RIF block diagram . . . 237
Figure 9. RIF access control principle . . . 239
Figure 10. Accesses to secure memories and peripherals . . . 240
Figure 11. Accesses to secure memories and peripherals . . . 240
Figure 12. RIF interrupt connection types . . . 241
Figure 13. Trusted domain chaining principle . . . 244
Figure 14. Wake-up CPU principle . . . 246
Figure 15. BSEC block diagram . . . 250
Figure 16. BSEC fuse mapping overview . . . 251
Figure 17. Device typical lifecycle . . . 253
Figure 18. RIMU block diagram (without inheritance) . . . 311
Figure 19. RIMU block diagram (with inheritance from RISUP programming) . . . 311
Figure 20. RISUP block diagram . . . 312
Figure 21. DAP and its RIMU . . . 314
Figure 22. Allocation of RISUP resources . . . 315
Figure 23. RISUP initialization principle . . . 317
Figure 24. RISAB block diagram . . . 331
Figure 25. RISAB region programming overview . . . 332
Figure 26. RISAB initialization principle . . . 333
Figure 27. RISAF block diagram (without DDRMCE) . . . 348
Figure 28. RISAF4 block diagram (with DDRMCE) . . . 349
Figure 29. RISAF region programming . . . 350
Figure 30. RISAF initialization principle . . . 353
Figure 31. RISAF region on-the-fly resizing . . . 354
Figure 32. IAC block diagram . . . 376
Figure 33. DDRSS top-level diagram . . . 384
Figure 34. ClkSwitcher architecture view . . . 386
Figure 35. AC-swizzling muxing stage . . . 388
Figure 36. Observability infrastructure organization . . . 391
Figure 37. Clock gating the DDRCTRL core clk in automatic self-refresh . . . 392
Figure 38. DDRCTRL block diagram . . . 406
Figure 39. Quasi-dynamic register programming . . . 414
Figure 40. DDRPHYC block diagram . . . 527
Figure 41. ACX4 component organization . . . 530
Figure 42. PHY clocking distribution . . . 531
Figure 43. DDRSS boot sequence . . . 532
Figure 44. DDRPERFM top view . . . 695
Figure 45. ICACHE block diagram . . . 737
Figure 46. ICACHE TAG and data memories functional view . . . 739
Figure 47. ICACHE remapping address mechanism . . . 742
Figure 48. DCACHE block diagram . . . 758

Figure 49. DCACHE TAG and data memories functional view . . . 761
Figure 50. Power control block diagram . . . 778
Figure 51. Power supply overview . . . 781
Figure 52. Device startup supply sequence . . . 784
Figure 53. Device Run2 supply sequence . . . 785
Figure 54. Device low-power Stop1 supply sequence . . . 786
Figure 55. Device low-power Stop2 supply sequence . . . 787
Figure 56. Device low-power LP-Stop1/LPLV-Stop1 supply sequence . . . 788
Figure 57. Device low-power LP-Stop2 supply sequence . . . 789
Figure 58. Device Standby mode power control . . . 790
Figure 59. Device Standby mode power control CPU2 wake-up alone . . . 790
Figure 60. Device VBAT mode power control . . . 791
Figure 61. VSW / Backup domain . . . 792
Figure 62. V_{DD} domain . . . 794
Figure 63. Power-on reset/power-down reset waveform . . . 796
Figure 64. BOR thresholds . . . 797
Figure 65. vddcore_ok reset thresholds . . . 797
Figure 66. vddcpu_ok reset thresholds . . . 798
Figure 67. PVD thresholds . . . 799
Figure 68. V_{08CAP} thresholds . . . 801
Figure 69. Temperature thresholds . . . 802
Figure 70. Power states and external regulator control . . . 811
Figure 71. Detailed state diagram of power control modes . . . 812
Figure 72. PWR_CPU_ON, PWR_ON and PWR_LP multiplexing . . . 827
Figure 73. RCC block diagram . . . 866
Figure 74. System reset circuit . . . 869
Figure 75. NRST reset pulse control . . . 873
Figure 76. NRSTC1MS reset pulse control . . . 874
Figure 77. Reset source registers . . . 877
Figure 78. Boot sequences versus system states . . . 880
Figure 79. HOLD_BOOT CPUx reset . . . 883
Figure 80. HOLD_BOOT state in legacy mode . . . 884
Figure 81. HOLD_BOOT state in wake-up single event mode . . . 884
Figure 82. HOLD_BOOT state in wake-up dual event mode . . . 885
Figure 83. Top-level clock tree . . . 887
Figure 84. HSE clock source . . . 889
Figure 85. HSE clock generation . . . 890
Figure 86. HSI calibration flow . . . 892
Figure 87. MSI calibration flow . . . 893
Figure 88. LSE clock generation . . . 894
Figure 89. HSE and LSE CSS function . . . 896
Figure 90. MCOy clock observation . . . 898
Figure 91. PLL block diagram . . . 899
Figure 92. PLL with SSMOD (spread-spectrum modulator) configuration . . . 899
Figure 93. Spread-spectrum modulation . . . 902
Figure 94. Flexclkgen block diagram . . . 905
Figure 95. Flexclkgen programming flow . . . 906
Figure 96. Root control module description . . . 909
Figure 97. Clock restore activation versus system mode . . . 914
Figure 98. Clock restore process activated after system Stop . . . 916
Figure 99. Wake-up from CStop . . . 917
Figure 100. Peripheral clock distribution for TIMs . . . 922

Figure 101. Peripheral clock distribution for ADCs . . . 923
Figure 102. Clock distribution for the DDR subsystem . . . 925
Figure 103. Entering and exiting self-refresh in ASR mode . . . 930
Figure 104. Entering and exiting self-refresh in HSR mode . . . 931
Figure 105. Allowed transitions between SSR, ASR and HSR modes . . . 932
Figure 106. Peripheral allocation example . . . 937
Figure 107. Peripheral clock enable logic details . . . 941
Figure 108. CPU1 low-power interface . . . 945
Figure 109. CPU1 PLL1 stop sequence . . . 945
Figure 110. FMC interface clocking . . . 947
Figure 111. OCTOSPI interface clocking . . . 948
Figure 112. SDMMC interface clocking . . . 950
Figure 113. Single GMAC clocks . . . 952
Figure 114. CA35SS interface clocks . . . 956
Figure 115. Debug and Trace external clocks . . . 958
Figure 116. Camera interface clocks . . . 959
Figure 117. Display subsystem clocking . . . 960
Figure 118. USBH and OTG ports clocking . . . 961
Figure 119. Security clocking . . . 964
Figure 120. Interrupt block diagram . . . 966
Figure 121. RCC RIF description . . . 972
Figure 122. IPCC block diagram . . . 1171
Figure 123. IPCC Simplex channel mode transfer timing . . . 1173
Figure 124. IPCC Simplex - Send procedure state diagram . . . 1173
Figure 125. IPCC Simplex - Receive procedure state diagram . . . 1174
Figure 126. IPCC Half-duplex channel mode transfer timing . . . 1175
Figure 127. IPCC Half-duplex - Send procedure state diagram . . . 1176
Figure 128. IPCC Half-duplex - Receive procedure state diagram . . . 1177
Figure 129. Basic structure of an I/O port bit . . . 1196
Figure 130. Input floating/pull-up/pull-down configurations . . . 1200
Figure 131. Output configuration . . . 1201
Figure 132. Alternate function configuration . . . 1201
Figure 133. Programmable analog configuration . . . 1202
Figure 134. I/O compensation cell control overview . . . 1233
Figure 135. HPDMA block diagram . . . 1328
Figure 136. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0) . . . 1329
Figure 137. HPDMA channel suspend and resume sequence . . . 1330
Figure 138. HPDMA channel abort and restart sequence . . . 1331
Figure 139. HPDMA channel terminate and drain FIFO sequence . . . 1332
Figure 140. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 1334
Figure 141. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 1335
Figure 142. HPDMA dynamic linked-list data structure of linear addressing channel x . . . 1336
Figure 143. HPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 1336
Figure 144. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0) . . . 1339
Figure 145. Inserting a LLIn with an auxiliary HPDMA channel y . . . 1341
Figure 146. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1) . . . 1343
Figure 147. Building LLIn+1: HPDMA dynamic linked-lists in link step mode . . . 1344
Figure 148. Replace with a new LLIn' in register file in link step mode . . . 1345
Figure 149. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 1346
Figure 150. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 1347

Figure 151. HPDMA channel execution and linked-list programming 1349
Figure 152. Programmed 2D addressing 1352
Figure 153. HPDMA arbitration policy 1367
Figure 154. Trigger hit, memorization and overrun waveform 1371
Figure 155. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel 1372
Figure 156. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel 1373
Figure 157. GIC block diagram 1460
Figure 158. EXTI block diagram 1505
Figure 159. Configurable event trigger logic CPU wake-up 1513
Figure 160. Direct event trigger logic CPU wake-up 1514
Figure 161. EXTI mux GPIO selection 1515
Figure 162. CRC calculation unit block diagram 1600
Figure 163. FMC interface block diagram 1607
Figure 164. FMC interface reset 1607
Figure 165. Octo-SPI interface block diagram 1609
Figure 166. OCTOSPI interface resets 1610
Figure 167. Programmable delay block diagram 1611
Figure 168. Flash configuration example: 2× 8-bit flash devices connected to the OCTOSPI interface 1614
Figure 169. Flash configuration example: dual quad mode. Two 4-bit devices are connected to the same Chip Select of the OCTOSPI interface . 1615
Figure 170. SD/eMMC interface block diagram 1620
Figure 171. SD/eMMC interface resets 1621
Figure 172. FMC block diagram 1629
Figure 173. Mode 1 read access waveforms 1648
Figure 174. Mode 1 write access waveforms 1648
Figure 175. Mode A read access waveforms 1650
Figure 176. Mode A write access waveforms 1651
Figure 177. Mode 2 and mode B read access waveforms 1653
Figure 178. Mode 2 write access waveforms 1653
Figure 179. Mode B write access waveforms 1654
Figure 180. Mode C read access waveforms 1656
Figure 181. Mode C write access waveforms 1656
Figure 182. Mode D read access waveforms 1658
Figure 183. Mode D write access waveforms 1659
Figure 184. Muxed read access waveforms 1661
Figure 185. Muxed write access waveforms 1662
Figure 186. Asynchronous wait during a read access waveforms 1665
Figure 187. Asynchronous wait during a write access waveforms 1665
Figure 188. Wait configuration waveforms 1667
Figure 189. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) 1668
Figure 190. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) 1670
Figure 191. NAND flash controller waveforms for common memory access 1687
Figure 192. NAND flash controller waveforms for TCLR and TAR timings 1687
Figure 193. Access to NAND flash memory 1689
Figure 194. NAND flash page (2 Kbytes) layout with BCH 1693
Figure 195. Example of 2 Kbyte page program sequence with 4-bit BCH 1694
Figure 196. Chained-page read command timings 1696
Figure 197. Example of 2-Kbyte page read sequence with 4-bit BCH 1697
Figure 198. OCTOSPI block diagram in octal configuration 1731

Figure 199. OCTOSPI block diagram in quad configuration . . . 1731
Figure 200. OCTOSPI block diagram in dual-quad configuration . . . 1732
Figure 201. SDR read command in octal configuration . . . 1733
Figure 202. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 1736
Figure 203. SDR write command in octo-SPI mode example . . . 1738
Figure 204. DTR write in octal-SPI mode (Macronix mode) example . . . 1739
Figure 205. Example of HyperBus read operation . . . 1741
Figure 206. HyperBus write operation with initial latency . . . 1742
Figure 207. HyperBus read operation with additional latency . . . 1742
Figure 208. HyperBus write operation with additional latency . . . 1743
Figure 209. HyperBus write operation with no latency (register write) . . . 1743
Figure 210. HyperBus read operation page crossing with latency . . . 1744
Figure 211. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 1751
Figure 212. OctaRAM read operation with reverse data ordering D1/D0 . . . 1751
Figure 213. NCS when CKMODE = 0 (T = CLK period) . . . 1757
Figure 214. NCS when CKMODE = 1 in SDR mode (T = CLK period) . . . 1757
Figure 215. NCS when CKMODE = 1 in DTR mode (T = CLK period) . . . 1757
Figure 216. NCS when CKMODE = 1 with an abort (T = CLK period) . . . 1758
Figure 217. Example of software control of two external memories . . . 1758
Figure 218. SDMMC “no response” and “no data” operations . . . 1791
Figure 219. SDMMC (multiple) block read operation . . . 1791
Figure 220. SDMMC (multiple) block write operation . . . 1792
Figure 221. SDMMC (sequential) stream read operation . . . 1792
Figure 222. SDMMC (sequential) stream write operation . . . 1792
Figure 223. SDMMC block diagram . . . 1794
Figure 224. SDMMC Command and data phase relation . . . 1796
Figure 225. Control unit . . . 1798
Figure 226. Command/response path . . . 1799
Figure 227. Command path state machine (CPSM) . . . 1800
Figure 228. Data path . . . 1806
Figure 229. DDR mode data packet clocking . . . 1807
Figure 230. DDR mode CRC status / boot acknowledgment clocking . . . 1807
Figure 231. Data path state machine (DPSM) . . . 1808
Figure 232. CLKMUX unit . . . 1819
Figure 233. Linked list structures . . . 1822
Figure 234. Asynchronous interrupt generation . . . 1825
Figure 235. Synchronous interrupt period data read . . . 1826
Figure 236. Synchronous interrupt period data write . . . 1826
Figure 237. Asynchronous interrupt period data read . . . 1827
Figure 238. Asynchronous interrupt period data write . . . 1828
Figure 239. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 1831
Figure 240. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 1831
Figure 241. Read Wait with SDMMC_CK < 50 MHz . . . 1832
Figure 242. Read Wait with SDMMC_CK > 50 MHz . . . 1832
Figure 243. CMD12 stream timing . . . 1835
Figure 244. CMD5 Sleep Awake procedure . . . 1837
Figure 245. Normal boot mode operation . . . 1839
Figure 246. Alternative boot mode operation . . . 1840
Figure 247. Command response R1b busy signaling . . . 1841
Figure 248. SDMMC state control . . . 1842
Figure 249. Card cycle power / power up diagram . . . 1843
Figure 250. Read block gap hardware flow timing . . . 1844

Figure 251. CMD11 signal voltage switch sequence . . . 1846
Figure 252. Voltage switch transceiver typical application . . . 1848
Figure 253. DLYB block diagram . . . 1878
Figure 254. ADC block diagram . . . 1886
Figure 255. ADC clock scheme . . . 1892
Figure 256. ADC1 connectivity . . . 1894
Figure 257. ADC2 connectivity . . . 1895
Figure 258. Enabling/disabling the ADC . . . 1900
Figure 259. Bulb mode timing diagram . . . 1903
Figure 260. Analog-to-digital conversion time . . . 1908
Figure 261. Stopping ongoing regular conversions . . . 1909
Figure 262. Stopping ongoing regular and injected conversions . . . 1909
Figure 263. Injected conversion latency during ongoing regular conversion . . . 1911
Figure 264. Single conversions of a sequence, software trigger . . . 1913
Figure 265. Continuous conversion of a sequence, software trigger . . . 1913
Figure 266. Single conversions of a sequence, hardware trigger . . . 1914
Figure 267. Continuous conversions of a sequence, hardware trigger . . . 1914
Figure 268. Right alignment (offset disabled, unsigned value) . . . 1916
Figure 269. Right alignment (offset enabled, signed value) . . . 1917
Figure 270. Left alignment (offset disabled, unsigned value) . . . 1917
Figure 271. Left alignment (offset enabled, signed value) . . . 1918
Figure 272. Example of overrun (OVRMOD = 0) . . . 1921
Figure 273. Example of overrun (OVRMOD = 1) . . . 1921
Figure 274. AUTDLY = 1, regular conversion in continuous mode, software trigger . . . 1925
Figure 275. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 1925
Figure 276. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 1926
Figure 277. AUTDLY = 1, regular continuous conversions interrupted by injected conversions . . . 1926
Figure 278. AUTDLY = 1 in autoinjection mode (JAUTO = 1) . . . 1927
Figure 279. Analog watchdog guarded area . . . 1928
Figure 280. ADC_AWDy_OUT signal generation (on all regular channels) . . . 1930
Figure 281. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 1931
Figure 282. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 1931
Figure 283. ADC_AWDy_OUT signal generation (on all injected channels) . . . 1932
Figure 284. 12-bit result oversampling with 10-bit right shift and rounding . . . 1933
Figure 285. Triggered regular oversampling mode (TROVS bit = 1) . . . 1935
Figure 286. Regular oversampling modes (4x ratio) . . . 1936
Figure 287. Regular and injected oversampling modes used simultaneously . . . 1937
Figure 288. Triggered regular oversampling with injection . . . 1937
Figure 289. Oversampling in autoinjection mode . . . 1938
Figure 290. VBAT channel block diagram . . . 1940
Figure 291. VREFINT channel block diagram . . . 1940
Figure 292. DTS block diagram . . . 1982
Figure 293. Serial data adapter . . . 1985
Figure 294. SDA data format . . . 1986
Figure 295. Falling alarms . . . 1991
Figure 296. Rising alarms . . . 1992
Figure 297. VREFBUF block diagram . . . 2023
Figure 298. MDF block diagram . . . 2031
Figure 299. SITFx overview . . . 2035
Figure 300. SPI timing example . . . 2036

Figure 301. Manchester timing example (SITFMOD = 11) . . . 2038
Figure 302. CKGEN overview . . . 2041
Figure 303. BSMX overview . . . 2043
Figure 304. SCD functional view . . . 2044
Figure 305. SCD timing example . . . 2044
Figure 306. DFLT overview . . . 2046
Figure 307. Programmable delay . . . 2047
Figure 308. CIC3 and CIC5 frequency response with decimation ratio = 32 . . . 2049
Figure 309. Reshape filter frequency response normalized (FRS / 2 = 1) . . . 2054
Figure 310. Out-of-limit detector thresholds . . . 2057
Figure 311. Trigger logic for DFLT and CKGEN . . . 2059
Figure 312. Asynchronous continuous mode (ACQMOD[2:0] = 0) . . . 2060
Figure 313. Asynchronous single-shot mode (ACQMOD[2:0] = 001) . . . 2061
Figure 314. Synchronous continuous mode (ACQMOD[2:0] = 010) . . . 2062
Figure 315. Synchronous single-shot mode (ACQMOD[2:0] = 011) . . . 2063
Figure 316. Window continuous mode (ACQMOD[2:0] = 100) . . . 2064
Figure 317. Snapshot mode example . . . 2066
Figure 318. Discard function example . . . 2067
Figure 319. Start sequence with DFLTEN, in continuous mode, audio configuration . . . 2068
Figure 320. Start sequence with trigger input, in continuous mode, motor configuration . . . 2069
Figure 321. Break interface simplified view . . . 2070
Figure 322. MDF_DFLTxDR data format . . . 2070
Figure 323. Data re-synchronization . . . 2071
Figure 324. Data transfer in interleaved-transfer mode . . . 2072
Figure 325. Data path for interleaved- and independent-transfer modes . . . 2074
Figure 326. Example of overflow and transfer to memory . . . 2075
Figure 327. MDF interrupt interface . . . 2078
Figure 328. Sensor connection examples . . . 2083
Figure 329. Global frequency response . . . 2084
Figure 330. Detailed frequency response . . . 2084
Figure 331. Simplified DFLT view with gain information . . . 2087
Figure 332. Camera subsystem block diagram . . . 2119
Figure 333. Camera subsystem clock diagram . . . 2120
Figure 334. DCMI block diagram . . . 2127
Figure 335. DCMI signal waveforms . . . 2128
Figure 336. Timing diagram . . . 2130
Figure 337. Frame capture waveforms in snapshot mode . . . 2132
Figure 338. Frame capture waveforms in continuous grab mode . . . 2133
Figure 339. Coordinates and size of the window after cropping . . . 2133
Figure 340. Data capture waveforms . . . 2134
Figure 341. Pixel raster scan order . . . 2135
Figure 342. DCMIPP overview . . . 2151
Figure 343. DCMIPP block diagram . . . 2155
Figure 344. VC flow processed by one pipe . . . 2172
Figure 345. Two exclusive VC flows processed by one reconfigured pipe . . . 2173
Figure 346. Two overlapping VC flows processed by two pipes . . . 2173
Figure 347. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes . . . 2174
Figure 348. Pipe0 (dump) architecture overview . . . 2177
Figure 349. Pipe1 ISP architecture view . . . 2181
Figure 350. Block diagram . . . 2204
Figure 351. Downsize with coverage filtering . . . 2207
Figure 352. Pipe2 architecture overview . . . 2211

Figure 353. Use case: Pipe0 (statistics), Pipe1 (pixels). . . . 2212
Figure 354. Statistics and RGB data for single virtual channel data flow . . . 2213
Figure 355. Three pipes within a single virtual channel in CSI2 mode . . . 2214
Figure 356. Handling interleaved packet . . . 2214
Figure 357. Handling interleaved packets across two pixel pipes . . . 2214
Figure 358. Interlaced video (based on single pixel pipe) . . . 2215
Figure 359. Interlaced video (based on two pixel pipes) . . . 2215
Figure 360. 3D sensors (top-bottom) processed by two pixel pipes . . . 2220
Figure 361. CSI-2 Host block diagram . . . 2359
Figure 362. CSI-2 Host architecture . . . 2360
Figure 363. CSI-2 Host clock domain . . . 2362
Figure 364. Lane merger input and output (one data lane) . . . 2363
Figure 365. Lane merger input and output (two data lanes) . . . 2363
Figure 366. Data lane mapping . . . 2363
Figure 367. Low-level protocol overview . . . 2364
Figure 368. CSI2- long-packet structure for a D-PHY physical layer . . . 2365
Figure 369. Short-packet structure based on D-PHY physical layer . . . 2365
Figure 370. DATA ID structure in short or long packet format . . . 2366
Figure 371. Data interleaving using data type . . . 2367
Figure 372. Data interleaving using virtual channels . . . 2368
Figure 373. RAW6 (BPP6) input and output format . . . 2369
Figure 374. Raw7 (BPP7) input and output format . . . 2370
Figure 375. Raw8 (BPP8) input and output format . . . 2371
Figure 376. Raw 10 (BPP10) input and output format . . . 2371
Figure 377. Raw12 (BPP12) input/output format . . . 2372
Figure 378. Raw14 (BPP14) input format . . . 2373
Figure 379. Raw16 (BPP16) input/output format . . . 2374
Figure 380. PSSI block diagram . . . 2420
Figure 381. Top-level block diagram . . . 2420
Figure 382. Data enable in receive mode waveform diagram (CKPOL=0) . . . 2424
Figure 383. Data enable waveform diagram in transmit mode (CKPOL=1) . . . 2424
Figure 384. Ready in receive mode waveform diagram (CKPOL=0) . . . 2425
Figure 385. Bidirectional PSSI_DE/PSSI_RDY waveform . . . 2426
Figure 386. Bidirectional PSSI_DE/PSSI_RDY connection diagram . . . 2426
Figure 387. Display subsystem block diagram . . . 2436
Figure 388. Display subsystem clock diagram . . . 2436
Figure 389. Display RIF peripheral ID . . . 2439
Figure 390. LTDC block diagram . . . 2444
Figure 391. Layer window programmable parameters . . . 2458
Figure 392. Blending two layers . . . 2458
Figure 393. LTDC synchronous timings . . . 2462
Figure 394. RNG block diagram . . . 2513
Figure 395. NIST SP800-90B entropy source model . . . 2514
Figure 396. RNG initialization overview . . . 2517
Figure 397. SAES usage in the device . . . 2533
Figure 398. SAES block diagram . . . 2534
Figure 399. Encryption/ decryption typical usage . . . 2536
Figure 400. Typical operation with authentication . . . 2538
Figure 401. Example of suspend mode management . . . 2539
Figure 402. ECB encryption . . . 2540
Figure 403. ECB decryption . . . 2540
Figure 404. CBC encryption . . . 2541

Figure 405. CBC decryption . . . 2541
Figure 406. Message construction in CTR mode . . . 2544
Figure 407. CTR encryption . . . 2545
Figure 408. Message construction in GCM . . . 2547
Figure 409. GCM authenticated encryption . . . 2548
Figure 410. Message construction in GMAC mode . . . 2551
Figure 411. GMAC authentication mode . . . 2551
Figure 412. Message construction in CCM mode . . . 2552
Figure 413. CCM mode authenticated encryption . . . 2554
Figure 414. Usage of wrapped keys in ECB and CBC modes . . . 2558
Figure 415. Operation with wrapped keys in CTR mode . . . 2560
Figure 416. 128-bit block construction according to the data type . . . 2564
Figure 417. Key protection mechanisms . . . 2566
Figure 418. CRYP block diagram . . . 2591
Figure 419. Encryption/ decryption typical usage . . . 2593
Figure 420. Typical operation with authentication . . . 2596
Figure 421. Example of suspend mode management . . . 2597
Figure 422. DES/TDES ECB encryption . . . 2598
Figure 423. DES/TDES ECB decryption . . . 2598
Figure 424. DES/TDES CBC encryption . . . 2599
Figure 425. DES/TDES CBC decryption . . . 2599
Figure 426. AES-ECB encryption . . . 2601
Figure 427. AES-ECB decryption . . . 2602
Figure 428. AES-CBC encryption . . . 2602
Figure 429. AES-CBC decryption . . . 2603
Figure 430. Message construction in CTR mode . . . 2605
Figure 431. AES-CTR encryption . . . 2606
Figure 432. Message construction in GCM . . . 2608
Figure 433. GCM authenticated encryption . . . 2609
Figure 434. Message construction in GMAC mode . . . 2613
Figure 435. GMAC authentication mode . . . 2613
Figure 436. Message construction in CCM mode . . . 2614
Figure 437. CCM mode authenticated encryption . . . 2616
Figure 438. 64-bit block construction according to the data type . . . 2621
Figure 439. 128-bit block construction according to the data type . . . 2623
Figure 440. HASH block diagram . . . 2648
Figure 441. Message data swapping feature . . . 2651
Figure 442. HASH suspend/resume mechanism . . . 2657
Figure 443. OTFDEC block diagram . . . 2674
Figure 444. Typical OTFDEC use in a SoC . . . 2675
Figure 445. AES CTR decryption flow . . . 2676
Figure 446. OTFDEC flow control overview (dual burst read request) . . . 2677
Figure 447. PKA block diagram . . . 2696
Figure 448. Advanced-control timer block diagram . . . 2731
Figure 449. Counter timing diagram with prescaler division change from 1 to 2 . . . 2737
Figure 450. Counter timing diagram with prescaler division change from 1 to 4 . . . 2738
Figure 451. Counter timing diagram, internal clock divided by 1 . . . 2739
Figure 452. Counter timing diagram, internal clock divided by 2 . . . 2740
Figure 453. Counter timing diagram, internal clock divided by 4 . . . 2740
Figure 454. Counter timing diagram, internal clock divided by N . . . 2741
Figure 455. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2741

Figure 456. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 2742
Figure 457. Counter timing diagram, internal clock divided by 1 . . . 2743
Figure 458. Counter timing diagram, internal clock divided by 2 . . . 2744
Figure 459. Counter timing diagram, internal clock divided by 4 . . . 2744
Figure 460. Counter timing diagram, internal clock divided by N . . . 2745
Figure 461. Counter timing diagram, update event when repetition counter is not used . . . 2745
Figure 462. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 2747
Figure 463. Counter timing diagram, internal clock divided by 2 . . . 2747
Figure 464. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 . . . 2748
Figure 465. Counter timing diagram, internal clock divided by N . . . 2748
Figure 466. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 2749
Figure 467. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2750
Figure 468. Update rate examples depending on mode and TIMx_RCR register settings . . . 2751
Figure 470. Control circuit in normal mode, internal clock divided by 1 . . . 2753
Figure 471. tim_ti2 external clock connection example . . . 2753
Figure 472. Control circuit in external clock mode 1 . . . 2754
Figure 473. External trigger input block . . . 2755
Figure 474. Control circuit in external clock mode 2 . . . 2756
Figure 475. Capture/compare channel (example: channel 1 input stage) . . . 2756
Figure 476. Capture/compare channel 1 main circuit . . . 2757
Figure 477. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 2757
Figure 478. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . 2758
Figure 479. PWM input mode timing . . . 2760
Figure 480. Output compare mode, toggle on tim_oc1 . . . 2762
Figure 481. Edge-aligned PWM waveforms (ARR = 8) . . . 2763
Figure 482. Center-aligned PWM waveforms (ARR = 8) . . . 2764
Figure 483. Dithering principle . . . 2765
Figure 484. Data format and register coding in dithering mode . . . 2766
Figure 485. PWM resolution vs frequency . . . 2767
Figure 486. PWM dithering pattern . . . 2768
Figure 487. Dithering effect on duty cycle in center-aligned PWM mode . . . 2769
Figure 488. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 2771
Figure 489. Combined PWM mode on channel 1 and 3 . . . 2772
Figure 490. 3-phase combined PWM signals with multiple trigger pulses per period . . . 2773
Figure 491. Complementary output with symmetrical dead-time insertion . . . 2774
Figure 492. Asymmetrical deadtime . . . 2775
Figure 493. Dead-time waveforms with delay greater than the negative pulse . . . 2775
Figure 494. Dead-time waveforms with delay greater than the positive pulse . . . 2775
Figure 495. Break and Break2 circuitry overview . . . 2778
Figure 496. Various output behavior in response to a break event on tim_brk (OSSI = 1) . . . 2780
Figure 497. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) . . . 2781
Figure 498. PWM output state following tim_brk assertion (OSSI = 0) . . . 2782
Figure 499. Output redirection (tim_brk2 request not represented) . . . 2783
Figure 500. Clearing TIMx tim_ocxref . . . 2784
Figure 501. 6-step generation, COM example (OSSR = 1) . . . 2785
Figure 502. Example of one pulse mode. . . . 2786
Figure 503. Retriggerable one-pulse mode . . . 2788
Figure 504. Pulse generator circuitry . . . 2788
Figure 505. Pulse generation on compare event, for edge-aligned and encoder modes . . . 2789
Figure 506. Extended pulsewidth in case of concurrent triggers . . . 2790
Figure 507. Example of counter operation in encoder interface mode. . . . 2792

Figure 508. Example of encoder interface mode with tim_ti1fp1 polarity inverted. 2792
Figure 509. Quadrature encoder counting modes 2793
Figure 510. Direction plus clock encoder mode 2794
Figure 511. Directional clock encoder mode (CC1P = CC2P = 0) 2794
Figure 512. Directional clock encoder mode (CC1P = CC2P = 1) 2795
Figure 513. Index gating options 2796
Figure 514. Jittered Index signals 2796
Figure 515. Index generation for IPOS[1:0] = 11 2797
Figure 516. Counter reading with index gated on channel A (IPOS[1:0] = 11) 2797
Figure 517. Counter reading with index ungated (IPOS[1:0] = 00) 2798
Figure 518. Counter reading with index gated on channel A and B 2798
Figure 519. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) 2799
Figure 520. Counter reset Narrow index pulse (closer view, ARR = 0x07) 2800
Figure 521. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) 2801
Figure 522. Directional index sensitivity 2801
Figure 523. Counter reset as function of FIDX bit setting 2802
Figure 524. Index blanking 2802
Figure 525. Index behavior in clock + direction mode, IPOS[0] = 1 2803
Figure 526. Index behavior in directional clock mode, IPOS[0] = 1 2803
Figure 527. State diagram for quadrature encoded signals 2804
Figure 528. Up-counting encoder error detection 2805
Figure 529. Down-counting encode error detection 2806
Figure 530. Encoder mode change with preload transferred on update (SMSPS = 0) 2807
Figure 531. Measuring time interval between edges on three signals 2808
Figure 532. Example of Hall sensor interface 2810
Figure 533. Control circuit in reset mode 2811
Figure 534. Control circuit in Gated mode 2812
Figure 535. Control circuit in trigger mode 2813
Figure 536. Control circuit in external clock mode 2 + trigger mode 2814
Figure 537. General-purpose timer block diagram 2876
Figure 538. Counter timing diagram with prescaler division change from 1 to 2 2881
Figure 539. Counter timing diagram with prescaler division change from 1 to 4 2882
Figure 540. Counter timing diagram, internal clock divided by 1 2883
Figure 541. Counter timing diagram, internal clock divided by 2 2883
Figure 542. Counter timing diagram, internal clock divided by 4 2884
Figure 543. Counter timing diagram, internal clock divided by N 2884
Figure 544. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2885
Figure 545. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded) 2886
Figure 546. Counter timing diagram, internal clock divided by 1 2887
Figure 547. Counter timing diagram, internal clock divided by 2 2888
Figure 548. Counter timing diagram, internal clock divided by 4 2888
Figure 549. Counter timing diagram, internal clock divided by N 2889
Figure 550. Counter timing diagram, Update event 2889
Figure 551. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 2891
Figure 552. Counter timing diagram, internal clock divided by 2 2891
Figure 553. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 2892
Figure 554. Counter timing diagram, internal clock divided by N 2892
Figure 555. Counter timing diagram, Update event with ARPE = 1 (counter underflow) 2893
Figure 556. Counter timing diagram, Update event with ARPE = 1 (counter overflow) 2894
Figure 557. Control circuit in normal mode, internal clock divided by 1 2895
Figure 558. tim_ti2 external clock connection example 2895
Figure 559. Control circuit in external clock mode 1 2896

Figure 560. External trigger input block . . . 2897
Figure 561. Control circuit in external clock mode 2 . . . 2898
Figure 562. Capture/compare channel (example: channel 1 input stage) . . . 2898
Figure 563. Capture/compare channel 1 main circuit . . . 2899
Figure 564. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) . . . 2899
Figure 565. PWM input mode timing . . . 2902
Figure 566. Output compare mode, toggle on tim_oc1 . . . 2904
Figure 567. Edge-aligned PWM waveforms (ARR = 8) . . . 2905
Figure 568. Center-aligned PWM waveforms (ARR = 8) . . . 2906
Figure 569. Dithering principle . . . 2907
Figure 570. Data format and register coding in dithering mode . . . 2908
Figure 571. PWM resolution vs frequency (16-bit mode) . . . 2909
Figure 572. PWM resolution vs frequency (32-bit mode) . . . 2909
Figure 573. PWM dithering pattern . . . 2910
Figure 574. Dithering effect on duty cycle in center-aligned PWM mode . . . 2911
Figure 575. Generation of two phase-shifted PWM signals with 50% duty cycle . . . 2913
Figure 576. Combined PWM mode on channels 1 and 3 . . . 2914
Figure 577. OCREF_CLR input selection multiplexer . . . 2915
Figure 578. Clearing TIMx tim_ocxref . . . 2915
Figure 579. Example of One-pulse mode . . . 2916
Figure 580. Retriggerable one-pulse mode . . . 2918
Figure 581. Pulse generator circuitry . . . 2919
Figure 582. Pulse generation on compare event, for edge-aligned and encoder modes . . . 2919
Figure 583. Extended pulse width in case of concurrent triggers . . . 2920
Figure 584. Example of counter operation in encoder interface mode . . . 2922
Figure 585. Example of encoder interface mode with tim_ti1fp1 polarity inverted . . . 2922
Figure 586. Quadrature encoder counting modes . . . 2923
Figure 587. Direction plus clock encoder mode . . . 2924
Figure 588. Directional clock encoder mode (CC1P = CC2P = 0) . . . 2925
Figure 589. Directional clock encoder mode (CC1P = CC2P = 1) . . . 2925
Figure 590. Index gating options . . . 2927
Figure 591. Jittered Index signals . . . 2927
Figure 592. Index generation for IPOS[1:0] = 11 . . . 2928
Figure 593. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 2928
Figure 594. Counter reading with index ungated (IPOS[1:0] = 00) . . . 2929
Figure 595. Counter reading with index gated on channel A and B . . . 2929
Figure 596. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 2930
Figure 597. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 2931
Figure 598. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 2932
Figure 599. Directional index sensitivity . . . 2932
Figure 600. Counter reset as function of FIDX bit setting . . . 2933
Figure 601. Index blanking . . . 2933
Figure 602. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 2934
Figure 603. Index behavior in directional clock mode, IPOS[0] = 1 . . . 2934
Figure 604. State diagram for quadrature encoded signals . . . 2935
Figure 605. Up-counting encoder error detection . . . 2936
Figure 606. Down-counting encode error detection . . . 2937
Figure 607. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 2938
Figure 608. Control circuit in reset mode . . . 2940
Figure 609. Control circuit in gated mode . . . 2941
Figure 610. Control circuit in trigger mode . . . 2941
Figure 611. Control circuit in external clock mode 2 + trigger mode . . . 2943

Figure 612. Master/Slave timer example . . . 2943
Figure 613. Master/slave connection example with 1 channel only timers . . . 2944
Figure 614. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 2945
Figure 615. Gating TIM_slv with Enable of TIM_mstr . . . 2946
Figure 616. Triggering TIM_slv with update of TIM_mstr. . . . 2947
Figure 617. Triggering TIM_slv with Enable of TIM_mstr . . . 2947
Figure 618. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 2948
Figure 619. Basic timer block diagram. . . . 2991
Figure 620. Control circuit in normal mode, internal clock divided by 1 . . . 2992
Figure 621. Counter timing diagram with prescaler division change from 1 to 2 . . . 2993
Figure 622. Counter timing diagram with prescaler division change from 1 to 4 . . . 2994
Figure 623. Counter timing diagram, internal clock divided by 1 . . . 2995
Figure 624. Counter timing diagram, internal clock divided by 2 . . . 2995
Figure 625. Counter timing diagram, internal clock divided by 4 . . . 2996
Figure 626. Counter timing diagram, internal clock divided by N. . . . 2996
Figure 627. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). . . . 2997
Figure 628. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). . . . 2998
Figure 629. Dithering principle . . . 2999
Figure 630. Data format and register coding in dithering mode . . . 2999
Figure 631. FCnt resolution vs frequency . . . 3000
Figure 632. PWM dithering pattern . . . 3000
Figure 633. General-purpose timer block diagram (TIM12). . . . 3016
Figure 634. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14) . . . 3017
Figure 635. Counter timing diagram with prescaler division change from 1 to 2 . . . 3020
Figure 636. Counter timing diagram with prescaler division change from 1 to 4 . . . 3020
Figure 637. Counter timing diagram, internal clock divided by 1 . . . 3021
Figure 638. Counter timing diagram, internal clock divided by 2 . . . 3022
Figure 639. Counter timing diagram, internal clock divided by 4 . . . 3022
Figure 640. Counter timing diagram, internal clock divided by N. . . . 3023
Figure 641. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). . . . 3023
Figure 642. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). . . . 3024
Figure 643. Control circuit in normal mode, internal clock divided by 1 . . . 3025
Figure 644. tim_ti2 external clock connection example . . . 3025
Figure 645. Control circuit in external clock mode 1 . . . 3026
Figure 646. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14) . . . 3027
Figure 647. Capture/compare channel 1 input stage (TIM12) . . . 3027
Figure 648. Capture/compare channel 1 main circuit . . . 3028
Figure 649. Output stage of capture/compare channel 1. . . . 3028
Figure 650. PWM input mode timing . . . 3030
Figure 651. Output compare mode, toggle on tim_oc1.. . . . 3032
Figure 652. Edge-aligned PWM waveforms (ARR = 8) . . . 3033
Figure 653. Dithering principle . . . 3034
Figure 654. Data format and register coding in dithering mode . . . 3034
Figure 655. PWM resolution vs frequency . . . 3035
Figure 656. PWM dithering pattern . . . 3036
Figure 657. Combined PWM mode on channel 1 and 2 . . . 3038
Figure 658. Example of one pulse mode . . . 3039
Figure 659. Retriggerable one pulse mode . . . 3040

Figure 660. Measuring time interval between edges on 2 signals . . . 3041
Figure 661. Control circuit in reset mode . . . 3042
Figure 662. Control circuit in gated mode . . . 3043
Figure 663. Control circuit in trigger mode . . . 3043
Figure 664. TIM15 block diagram . . . 3086
Figure 665. TIM16/TIM17 block diagram . . . 3087
Figure 666. Counter timing diagram with prescaler division change from 1 to 2 . . . 3091
Figure 667. Counter timing diagram with prescaler division change from 1 to 4 . . . 3092
Figure 668. Counter timing diagram, internal clock divided by 1 . . . 3093
Figure 669. Counter timing diagram, internal clock divided by 2 . . . 3094
Figure 670. Counter timing diagram, internal clock divided by 4 . . . 3094
Figure 671. Counter timing diagram, internal clock divided by N . . . 3095
Figure 672. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 3095
Figure 673. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 3096
Figure 674. Update rate examples depending on mode and TIMx_RCR register settings . . . 3097
Figure 675. Control circuit in normal mode, internal clock divided by 1 . . . 3098
Figure 676. tim_ti2 external clock connection example . . . 3098
Figure 677. Control circuit in external clock mode 1 . . . 3099
Figure 678. Capture/compare channel (example: channel 1 input stage) . . . 3100
Figure 679. Capture/compare channel 1 main circuit . . . 3100
Figure 680. Output stage of capture/compare channel (channel 1) . . . 3101
Figure 681. Output stage of capture/compare channel (channel 2 for TIM15) . . . 3101
Figure 682. PWM input mode timing . . . 3104
Figure 683. Output compare mode, toggle on tim_oc1 . . . 3106
Figure 684. Edge-aligned PWM waveforms (ARR = 8) . . . 3107
Figure 685. Dithering principle . . . 3108
Figure 686. Data format and register coding in dithering mode . . . 3108
Figure 687. PWM resolution vs frequency . . . 3109
Figure 688. PWM dithering pattern . . . 3110
Figure 689. Combined PWM mode on channel 1 and 2 . . . 3112
Figure 690. Complementary output with symmetrical dead-time insertion . . . 3113
Figure 691. Asymmetrical deadtime . . . 3114
Figure 692. Dead-time waveforms with delay greater than the negative pulse . . . 3114
Figure 693. Dead-time waveforms with delay greater than the positive pulse . . . 3114
Figure 694. Break circuitry overview . . . 3116
Figure 695. Output behavior in response to a break event on tim_brk . . . 3118
Figure 696. Output redirection . . . 3120
Figure 697. tim_ocref_clr input selection multiplexer . . . 3121
Figure 698. 6-step generation, COM example (OSSR = 1) . . . 3122
Figure 699. Example of one pulse mode . . . 3123
Figure 700. Retriggerable one pulse mode . . . 3125
Figure 701. Measuring time interval between edges on 2 signals . . . 3125
Figure 702. Control circuit in reset mode . . . 3126
Figure 703. Control circuit in gated mode . . . 3127
Figure 704. Control circuit in trigger mode . . . 3128
Figure 705. LPTIM1/2/3/4 block diagram(1) . . . 3199
Figure 706. LPTIM5 block diagram(1) . . . 3200
Figure 707. Glitch filter timing diagram . . . 3204
Figure 708. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 3206

Figure 709. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 3206
Figure 710. LPTIM output waveform, Continuous counting mode configuration . . . 3207
Figure 711. Waveform generation . . . 3208
Figure 712. Encoder mode counting sequence . . . 3212
Figure 713. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 3213
Figure 714. Capture/compare input stage (channel 1) . . . 3214
Figure 715. Capture/compare output stage (channel 1) . . . 3214
Figure 716. Edge-aligned PWM mode (PRELOAD = 1) . . . 3216
Figure 717. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 3217
Figure 718. PWM mode with immediate update versus preloaded update . . . 3218
Figure 719. STGEN block diagram . . . 3253
Figure 720. Top view of independent watchdogs . . . 3270
Figure 721. Windows watchdogs top view . . . 3270
Figure 722. Independent watchdog block diagram . . . 3272
Figure 723. Reset timing due to timeout . . . 3274
Figure 724. Reset timing due to refresh in the not allowed area . . . 3275
Figure 725. Changing PR, RL, and performing a refresh(1) . . . 3276
Figure 726. Window comparator update(1) . . . 3277
Figure 727. Independent watchdog interrupt timing diagram . . . 3279
Figure 728. Early wake-up comparator update(1) . . . 3280
Figure 729. Watchdog block diagram . . . 3290
Figure 730. Window watchdog timing diagram . . . 3292
Figure 731. RTC block diagram . . . 3300
Figure 732. TAMP block diagram . . . 3366
Figure 733. Backup registers protection zones . . . 3371
Figure 734. Access rights for TAMP resources according to the CID . . . 3375
Figure 735. Tamper sampling with precharge pulse . . . 3379
Figure 736. Low level detection with precharge and filtering . . . 3380
Figure 737. Active tamper filtering . . . 3383
Figure 738. Block diagram . . . 3428
Figure 739. I²C-bus protocol . . . 3432
Figure 740. Setup and hold timings . . . 3434
Figure 741. I2C initialization flow . . . 3436
Figure 742. Data reception . . . 3437
Figure 743. Data transmission . . . 3438
Figure 744. Target initialization flow . . . 3441
Figure 745. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 3443
Figure 746. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 3444
Figure 747. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 3445
Figure 748. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 3446
Figure 749. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 3447
Figure 750. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 3447
Figure 751. Controller clock generation . . . 3449
Figure 752. Controller initialization flow . . . 3451
Figure 753. 10-bit address read access with HEAD10R = 0 . . . 3451
Figure 754. 10-bit address read access with HEAD10R = 1 . . . 3452
Figure 755. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 3453
Figure 756. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 3454
Figure 757. Transfer bus diagrams for I2C controller transmitter

(mandatory events only) . . . 3455
Figure 758. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 3457
Figure 759. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 3458
Figure 760. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 3459
Figure 761. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 3463
Figure 762. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 3466
Figure 763. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 3467
Figure 764. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 3468
Figure 765. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 3469
Figure 766. Bus transfer diagrams for SMBus controller transmitter . . . 3470
Figure 767. Bus transfer diagrams for SMBus controller receiver . . . 3472
Figure 768. I3C block diagram . . . 3501
Figure 769. I3C (primary) controller state and programming sequence diagram . . . 3506
Figure 770. I3C target state and programing sequence diagram . . . 3511
Figure 771. I3C CCC messages, as controller . . . 3525
Figure 772. I3C broadcast ENTDAA CCC, as controller . . . 3526
Figure 773. I3C broadcast, direct read and direct write RSTACT CCC, as controller . . . 3527
Figure 774. I3C CCC messages, as target . . . 3529
Figure 775. I3C broadcast ENTDAA CCC, as target . . . 3530
Figure 776. I3C broadcast DEFTGTS CCC, as target . . . 3531
Figure 777. I3C broadcast DEFGRPA CCC, as target . . . 3532
Figure 778. I3C private read/write messages, as controller . . . 3534
Figure 779. I3C private read/write messages, as target . . . 3535
Figure 780. Legacy I2C read/write messages, as controller . . . 3536
Figure 781. IBI transfer, as controller/target . . . 3537
Figure 782. Hot-join request transfer, as controller/target . . . 3539
Figure 783. Controller-role request transfer, as controller/target . . . 3540
Figure 784. C-FIFO management, as controller . . . 3541
Figure 785. TX-FIFO management, as controller . . . 3543
Figure 786. RX-FIFO management, as controller . . . 3545
Figure 787. S-FIFO management, as controller . . . 3548
Figure 788. RX-FIFO management, as target on the I3C bus . . . 3549
Figure 789. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus . . . 3551
Figure 790. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target . . . 3553
Figure 791. USART block diagram . . . 3613
Figure 792. Word length programming . . . 3617
Figure 793. Configurable stop bits . . . 3619
Figure 794. TC/TXE behavior when transmitting . . . 3621
Figure 795. Start bit detection when oversampling by 16 or 8 . . . 3622
Figure 796. usart_ker_ck clock divider block diagram . . . 3625
Figure 797. Data sampling when oversampling by 16 . . . 3626
Figure 798. Data sampling when oversampling by 8 . . . 3627
Figure 799. Mute mode using Idle line detection . . . 3634
Figure 800. Mute mode using address mark detection . . . 3635
Figure 801. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 3638
Figure 802. Break detection in LIN mode vs. Framing error detection . . . 3639
Figure 803. USART example of synchronous master transmission . . . 3640
Figure 804. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 3640
Figure 805. USART data clock timing diagram in synchronous master mode

(M bits = 01) . . . 3641
Figure 806. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 3642
Figure 807. ISO 7816-3 asynchronous protocol . . . 3644
Figure 808. Parity error detection using the 1.5 stop bits . . . 3646
Figure 809. IrDA SIR ENDEC block diagram . . . 3650
Figure 810. IrDA data modulation (3/16) - normal mode . . . 3650
Figure 811. Transmission using DMA . . . 3652
Figure 812. Reception using DMA . . . 3653
Figure 813. Hardware flow control between two USARTs . . . 3653
Figure 814. RS232 RTS flow control . . . 3654
Figure 815. RS232 CTS flow control . . . 3655
Figure 816. LPUART block diagram . . . 3705
Figure 817. LPUART word length programming . . . 3709
Figure 818. Configurable stop bits . . . 3711
Figure 819. TC/TXE behavior when transmitting . . . 3713
Figure 820. lpuart_ker_ck clock divider block diagram . . . 3717
Figure 821. Mute mode using Idle line detection . . . 3721
Figure 822. Mute mode using address mark detection . . . 3722
Figure 823. Transmission using DMA . . . 3724
Figure 824. Reception using DMA . . . 3725
Figure 825. Hardware flow control between an LPUART and another device . . . 3726
Figure 826. RS232 CTS flow control . . . 3727
Figure 827. SPI/I2S block diagram . . . 3762
Figure 828. Full-duplex single master/ single slave application . . . 3766
Figure 829. Half-duplex single master/ single slave application . . . 3766
Figure 830. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 3767
Figure 831. Master and three independent slaves connected in star topology . . . 3769
Figure 832. Master and three slaves connected in circular (daisy chain) topology . . . 3771
Figure 833. Multimaster application . . . 3772
Figure 834. Scheme of NSS control logic . . . 3774
Figure 835. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 3774
Figure 836. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 3775
Figure 837. Data clock timing diagram . . . 3777
Figure 838. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 3778
Figure 839. TI mode transfer . . . 3788
Figure 840. Optional configurations of the slave behavior when an underrun condition is detected . 3790
Figure 841. Waveform examples . . . 3798
Figure 842. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 3799
Figure 843. I2S Philips standard waveforms . . . 3799
Figure 844. Master MSB-justified 16- or 32-bit full-accuracy length . . . 3800
Figure 845. Master MSB-justified 16- or 24-bit data length . . . 3800
Figure 846. Slave MSB-justified 16-, 24- or 32-bit data length . . . 3801
Figure 847. LSB-justified 16 or 24-bit data length . . . 3801
Figure 848. Master PCM when the frame length is equal the data length . . . 3802
Figure 849. Master PCM standard waveforms (16 or 24-bit data length) . . . 3802
Figure 850. Slave PCM waveforms . . . 3803
Figure 851. Startup sequence, I2S Philips standard, master . . . 3806
Figure 852. Startup sequence, I2S Philips standard, slave . . . 3807
Figure 853. Stop sequence, I2S Philips standard, master . . . 3807
Figure 854. I^2S clock generator architecture . . . 3808

Figure 855. Data Format . . . 3810
Figure 856. Handling of underrun situation . . . 3812
Figure 857. Handling of overrun situation . . . 3813
Figure 858. Frame error detection, with FIXCH = 0 . . . 3814
Figure 859. Frame error detection, with FIXCH = 1 . . . 3814
Figure 860. SAI functional block diagram . . . 3844
Figure 861. Audio frame . . . 3848
Figure 862. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 3850
Figure 863. FS role is start of frame (FSDEF = 0) . . . 3851
Figure 864. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 3852
Figure 865. First bit offset . . . 3852
Figure 866. Audio block clock generator overview . . . 3854
Figure 867. PDM typical connection and timing . . . 3858
Figure 868. Detailed PDM interface block diagram . . . 3859
Figure 869. Start-up sequence . . . 3860
Figure 870. SAI_ADR format in TDM mode, 32-bit slot width . . . 3861
Figure 871. SAI_ADR format in TDM mode, 16-bit slot width . . . 3862
Figure 872. SAI_ADR format in TDM mode, 8-bit slot width . . . 3863
Figure 873. AC’97 audio frame . . . 3866
Figure 874. Example of typical AC’97 configuration on devices featuring at least two embedded SAIs (three external AC’97 decoders) . . . 3867
Figure 875. SPDIF format . . . 3868
Figure 876. SAI_xDR register ordering . . . 3869
Figure 877. Data companding hardware in an audio block in the SAI . . . 3872
Figure 878. Tristate strategy on SD output line on an inactive slot . . . 3874
Figure 879. Tristate on output data line in a protocol like I2S . . . 3875
Figure 880. Overrun detection error . . . 3876
Figure 881. FIFO underrun event . . . 3876
Figure 882. SPDIFRX block diagram . . . 3913
Figure 883. S/PDIF sub-frame format . . . 3914
Figure 884. S/PDIF block format . . . 3915
Figure 885. S/PDIF preambles . . . 3915
Figure 886. Channel coding example . . . 3916
Figure 887. SPDIFRX decoder . . . 3917
Figure 888. Noise filtering and edge detection . . . 3917
Figure 889. Thresholds . . . 3919
Figure 890. Synchronization flowchart . . . 3921
Figure 891. Synchronization process scheduling . . . 3922
Figure 892. SPDIFRX states . . . 3923
Figure 893. SPDIFRX_FMTx_DR register format . . . 3925
Figure 894. Channel/user data format . . . 3926
Figure 895. S/PDIF overrun error when RXSTEO = 0 . . . 3928
Figure 896. S/PDIF overrun error when RXSTEO = 1 . . . 3929
Figure 897. SPDIFRX interface interrupt mapping diagram . . . 3932
Figure 898. CAN subsystem . . . 3950
Figure 899. FDCAN block diagram . . . 3952
Figure 900. Transceiver delay measurement . . . 3957
Figure 901. Pin control in bus monitoring mode . . . 3959
Figure 902. Pin control in loop back mode . . . 3961
Figure 903. CAN error state diagram . . . 3962
Figure 904. Message RAM configuration . . . 3963
Figure 905. Standard message ID filter path . . . 3966

Figure 906. Extended message ID filter path . . . 3967
Figure 907. Example of mixed configuration dedicated Tx buffers / Tx FIFO . . . 3973
Figure 908. Example of mixed configuration dedicated Tx buffers / Tx queue . . . 3973
Figure 909. Bit timing . . . 3975
Figure 910. Bypass operation . . . 3977
Figure 911. FSM calibration . . . 3978
Figure 912. Cycle time and global time synchronization . . . 3993
Figure 913. TTCAN level 0 and level 2 drift compensation . . . 3994
Figure 914. Level 0 schedule synchronization state machine . . . 4001
Figure 915. Level 0 master to slave relation . . . 4002
Figure 916. USB sub-system overview . . . 4079
Figure 917. USB2 host high-speed port block diagram . . . 4082
Figure 918. USB2 host high-speed port reset . . . 4084
Figure 919. USB2 OTG high-speed port block diagram . . . 4090
Figure 920. USB2 OTG high-speed port resets . . . 4091
Figure 921. Multiplexing of debug signals in the USB Sub-system . . . 4097
Figure 922. OTG high-speed block diagram . . . 4102
Figure 923. OTG A-B device connection . . . 4104
Figure 924. OTG peripheral-only connection . . . 4105
Figure 925. OTG host-only connection . . . 4109
Figure 926. SOF connectivity (SOF trigger output to TIM and ITR1 connection) . . . 4112
Figure 927. Updating OTG_HFIR dynamically (RLDCTRL = 1) . . . 4114
Figure 928. Device-mode FIFO address mapping and AHB FIFO access mapping . . . 4115
Figure 929. Host-mode FIFO address mapping and AHB FIFO access mapping . . . 4116
Figure 930. Interrupt hierarchy . . . 4120
Figure 931. Transmit FIFO write task . . . 4213
Figure 932. Receive FIFO read task . . . 4214
Figure 933. Normal bulk/control OUT/SETUP . . . 4215
Figure 934. Bulk/control IN transactions . . . 4219
Figure 935. Normal interrupt OUT . . . 4222
Figure 936. Normal interrupt IN . . . 4227
Figure 937. Isochronous OUT transactions . . . 4229
Figure 938. Isochronous IN transactions . . . 4232
Figure 939. Normal bulk/control OUT/SETUP transactions - DMA . . . 4234
Figure 940. Normal bulk/control IN transaction - DMA . . . 4236
Figure 941. Normal interrupt OUT transactions - DMA mode . . . 4237
Figure 942. Normal interrupt IN transactions - DMA mode . . . 4238
Figure 943. Normal isochronous OUT transaction - DMA mode . . . 4239
Figure 944. Normal isochronous IN transactions - DMA mode . . . 4240
Figure 945. Receive FIFO packet read . . . 4246
Figure 946. Processing a SETUP packet . . . 4248
Figure 947. Bulk OUT transaction . . . 4255
Figure 948. TRDT max timing case . . . 4264
Figure 949. USBH block diagram . . . 4267
Figure 950. Ethernet subsystem block diagram . . . 4282
Figure 951. ETHSS external signals (simplified view) . . . 4283
Figure 952. Ethernet subsystem internal signals . . . 4284
Figure 953. Ethernet high-level block diagram . . . 4294
Figure 954. DMA transmission flow (standard mode) . . . 4297
Figure 955. DMA transmission flow (OSP mode) . . . 4299
Figure 956. Receive DMA flow . . . 4301
Figure 957. Overview of MAC transmission flow . . . 4304

Figure 958. MAC reception flow 4306
Figure 959. Packet filtering sequence 4323
Figure 960. Networked time synchronization 4337
Figure 961. Propagation delay calculation in clocks supporting peer-to-peer path correction 4338
Figure 962. System time update using fine correction method 4348
Figure 963. Time aware shaper implementation for traffic scheduling 4360
Figure 964. Implementing a guard band to avoid delays due to interfering frames 4361
Figure 965. GCL governing gate close and open events block diagram (from IEEE 802.1 Qbv specifications) 4362
Figure 966. GCL and associated registers - BaseTime and CycleTime 4366
Figure 967. GCL and associated registers - BaseTime and CycleTime list execution time > CycleTime 4367
Figure 968. Switching to a new Configuration that is Aligned with the Existing Configuration 4368
Figure 969. Switching to the New List by Truncating the Current List 4370
Figure 970. Switching to New List By Extending the Current List 4371
Figure 971. mPacket formats 4375
Figure 972. MAC data Alignment Feature 4380
Figure 973. Time-based scheduler implementation 4388
Figure 974. Time-based scheduling flow 4389
Figure 975. TCP segmentation offload overview 4395
Figure 976. TCP segmentation offload flow 4396
Figure 977. Header and payload fields of segmented packets 4399
Figure 978. Supported PHY interfaces 4411
Figure 979. SMA Interface block 4412
Figure 980. MDIO packet structure (Clause 45) 4413
Figure 981. MDIO packet structure (Clause 22) 4414
Figure 982. SMA write operation flow 4415
Figure 983. Write data packet 4416
Figure 984. Read data packet 4416
Figure 985. Media independent interface (MII) signals 4418
Figure 986. RMII block diagram 4420
Figure 987. Transmission bit order 4421
Figure 988. Receive bit order 4422
Figure 989. RGMII block diagram and interface signals 4423
Figure 990. LPI transitions (Transmit, 100 Mbds) 4432
Figure 991. LPI Tx clock gating (when LPITCSE = 1) 4433
Figure 992. LPI transitions (receive, 100 Mbps) 4434
Figure 993. Descriptor ring structure 4466
Figure 994. DMA descriptor ring 4467
Figure 995. Descriptor tail pointer example 1 4468
Figure 996. Descriptor tail pointer example 2 4468
Figure 997. Transmit descriptor (read format) 4469
Figure 998. Transmit descriptor write-back format 4474
Figure 999. Transmit context descriptor format 4479
Figure 1000. Receive normal descriptor (read format) 4482
Figure 1001. Receive normal descriptor (write-back format) 4485
Figure 1002. Receive context descriptor 4493
Figure 1003. Enhanced normal descriptor (read format) 4495
Figure 1004. Enhanced normal descriptor (write format) 4496
Figure 1005. Enhanced context descriptor (read format) 4497
Figure 1006. Enhanced context descriptor (write format) 4498

Figure 1007. Generation of ETH_DMAISR flags 4523
Figure 1008. Hardware debug port block diagram 4737
Figure 1009. SERF overview 4745
Figure 1010. AXI SERD block diagram 4747
Figure 1011. AHB SERD for IP block diagram 4748
Figure 1012. AHB SERD for ICN block diagram 4749
Figure 1013. AHB SERDAB for AHB2APB block diagram 4750
Figure 1014. SERC block diagram 4750
Figure 1015. SERF with an RISUP 4751
Figure 1016. Block diagram of debug support infrastructure 4767
Figure 1017. Debug clock domains 4769
Figure 1018. Debug and access port connections 4775
Figure 1019. CoreSight™ components topology 4779
Figure 1020. CoreSight timestamp distribution 4780
Figure 1021. Embedded cross trigger 4782
Figure 1022. Mapping trigger inputs to outputs 4786
Figure 1023. Cross trigger configuration example 4787
Figure 1024. System trace macrocell 4791
Figure 1025. Cortex-M33 CoreSight topology 4822

1 Documentation conventions

1.1 General information

The STM32MP21xx devices embed an Arm®[a] Cortex®-M33, and an Arm® Cortex®-A35 core.

arm

1.2 List of abbreviations for registers

The following abbreviations[b] are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- The product integrates two debug ports:
  - JTAG debug port (**JTAG-DP**) provides a 5-pin standard interface based on the Joint Test Action Group (JTAG) protocol.
  - SWD debug port (**SWD-DP**) provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol.
- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Double word**: data of 64-bit length.
- **Option bytes**: product configuration bits stored in internal fuses.
- **OTP**: one-time programming (fuses).
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.
- **AXI**: Advanced extensible interface protocol.
- **ECC**: error code correction.
- **DMA**: direct memory access.
- **AHBS**: AHB slave bus.
- **MLAHB**: Multi-Layer AHB interconnect matrix.
- **STNoC**: ST network on-chip interconnect.

1.5 Availability of peripherals

For availability of peripherals and their number across all devices, refer to the particular device datasheet.

2 Memory and bus architecture

2.1 System architecture

In the whole document:

- The Cortex-A35 is also called CPU1 or MPU.
- The Cortex-M33 is also called CPU2 or MCU.

2.1.1 Bus architecture

The bus architecture is organized around two interconnect matrices, which operate in different frequency domains, as described in *Figure 1*.

- one high-speed multi-frequency domain, AXI compliant STNoC interconnect:

  This high-performance interconnect is mainly used by the CPU1 subsystem and the following high-bandwidth masters (this part of the system is called MPU-side in the rest of the document): USBH EHCI, USBH OHCI, OTG, ETH1/2, SDMMC1/2/3, HPDMA1/2/3 (for their AXI ports), LTDC, DCMIPP.

  The CPU2 traffic directed to these targets ends up also in this interconnect.

  It manages accesses to:

  – external memories DDR (through DDRCTRL), NAND/NOR (through FMC)
  – main system internal memories (SYSRAM, ROM)
  – main cluster of security peripherals through the AHB bus
  – some low-speed communication peripherals through the APB4 bus
  – control and configuration of the high-bandwidth masters
  – STM (system trace macrocell)

Note: Boot ROM and GIC are only accessible to the CPU1 through an intermediate AXI interconnect downstream of the CPU1 (see Figure 1).

- one multi-layer AHB interconnect (this part of the system is called MCU-side in the following). It is itself spread in two MLAHBs (MLAHB300 and MLAHB150, where 150 and 300 refer to the maximum clock frequency of the interconnect).

  This low-latency interconnect receives traffic from the CPU2 subsystem and associated masters: HPDMA1/2/3 (for their AHB ports). High-bandwidth initiators whose traffics are directed towards peripherals and MCU-side memories (SRAM, OCTOSPI) end up in these MLAHB as well.

  MLAHB300 manages accesses to:

  – CPU2 dedicated SRAM1
  – CPU2 dedicated retention memory (RETRAM)
  – external memories NAND/NOR through OCTOSPI
  – connection to MLAHB150 through the frequency downsizer

MLAHB150 manages accesses to:

- most timers and low-speed communication peripherals through AHB2, AHB6 and APB1/2/5 bus
- secondary cluster of security peripherals and camera interface through AHB3/2 bus
- resources dedicated to communication with CPU1 cluster (mailbox through the AHB2 bus)
- main infrastructure blocks as clock generation, power control, system configuration, security configuration, and miscellaneous peripherals through the AHB4 bus

- AXI and multi-layer AHB matrices are connected to each other through dedicated data paths in both directions.
- One STNoC master port has access to MLAHB300 slaves, allowing access from the CPU1 to SRAM1, RETRAM, and OCTOSPI.
- One STNoC master port has access to MLAHB150 slaves, allowing access from the CPU1 to all peripherals.
- One MLAHB300 master port has access to STNoC through an AHB2AXI converter, allowing access from the CPU2 to DDR, SYSRAM, STM and FMC.
- One MLAHB150 master port has access to STNoC through an AHB2AXI converter, allowing access from the CPU2 to MPU-side peripherals.
- CPU2 has no access to the ROM and GIC.

The bus architecture diagram in *Figure 1* describes the interconnect for STM32MP21xx devices.

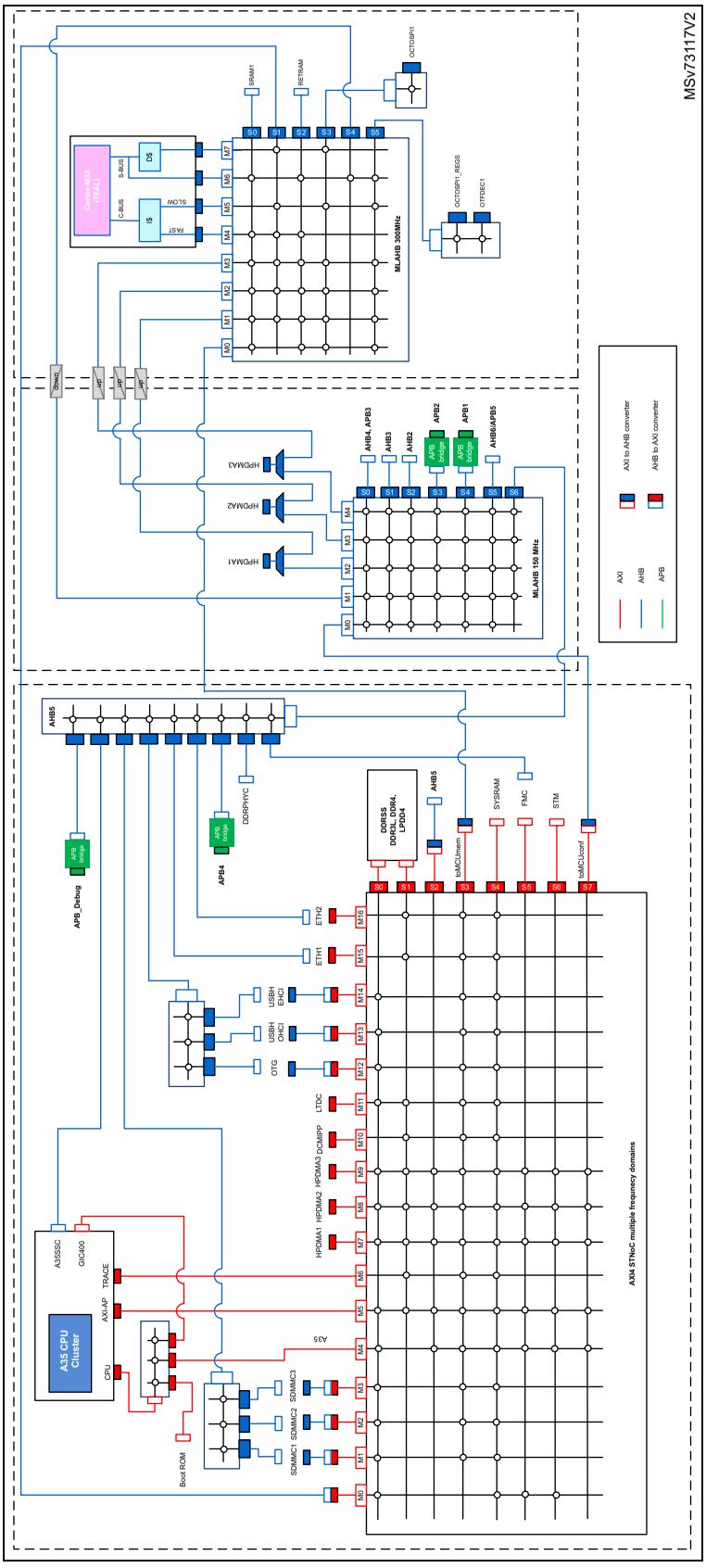


Figure 1. STM32MP21xx interconnect top view

2.1.2 Memory map organization

Introduction

For the detailed mapping of available memory and register areas, refer to *Section 2.4: Memory organization*.

The following paragraphs describe the various resources in the system.

Embedded memories

- **Boot ROM**

  The 192-Kbyte embedded ROM is dedicated to the CPU1 subsystem for boot code execution. This memory and the associated memory controller operate at 400 MHz typical.

- **SYSRAM**

  The 256-Kbyte embedded SYSRAM is preferably used by the CPU1 subsystem for code execution and low-latency data management. The CPU2 has access too. The SRAM itself is 0 wait-state. Some latency is added when data are propagated through the AXI interconnect (access from MPU-side). This memory and the associated memory controller operate at 400 MHz typical.

- **MCU SRAM1**

  The SRAM1 is a 64-Kbyte RAM preferably used by the CPU2 for the code execution, low-latency data management, and data storage for exchange with the CPU1 subsystem. The SRAM1 is 0 wait-state. This memory and the associated memory controller operate at 300 MHz typical.

- **Retention RAM (RETRAM)**

  The RETRAM is a 128-Kbyte RAM with specific supply providing data retention in low-power modes (including Standby and V_{BAT} modes). The RETRAM is 0 wait-state. Some latency is added when data are propagated through the AXI interconnect (access from MPU-side). This memory and the associated memory controller operate at 300 MHz typical.

- **Backup RAM (BKPSRAM)**

  The backup RAM is an 8-Kbyte static RAM with dual-rail supply providing data backup for the whole system, whatever the power mode, with standard latency. This memory and the associated memory controller operate at 150 MHz typical.

External memory support

- **DDR3L/4 LPDDR4 controller (DDRCTRL)**

  The DDR supported types are LPDDR4, DDR3L, and DDR4. Regarding external memory, 16-bit data width is supported, while internally, the DDRCTRL is connected to 128-bit data width interconnect. The maximum supported size is 4 Gbytes (DDR4). The DDRCTRL is accessible through two AXI data ports, which have same characteristics. However, only a transaction going through port 1 can be AES encrypted. The secure traffic is then default routed to the DDRSS XPI0 while non-secure traffic is routed towards the DDRSS XPI1.

  The following table presents the routing of the STNoC ports towards the DDR subsystem.

Table 1. STNoC routing of the DDR traffic

Initiator	DDRSS XPI0	DDRSS XPI1	Comment
CPU1	All traffic	None	-
AXI-AP (DebugSS)	Secure traffic	Non-secure traffic	-
TRACE	All traffic	None	-
DCMIPP	None	All traffic	-
LTDC	Secure traffic	Non secure traffic	-
OTG	None	All traffic	-
ETH1/2	None	All traffic	-
USBH (EHCI/OHCI)	All traffic	None	-
SDMMC1/2/3	All traffic	None	-
HPDMA1/2/3	Secure traffic	Non secure traffic	Relative to HPDMA AXI ports
MCU1	All traffic	None	Traffic from CPU2

More information on memory characteristics and controller parametrization is available in *Section 13: DDR3L/DDR4/LPDDR4 controller (DDRCTRL)* and *Section 14: DDR physical interface control (DDRPHYC)*. The DDRCTRL operates at up to ddr_ck = 400 MHz. The external memory operates at up to 1600 MHz (800 MHz dual data rate) depending on the system configuration.

- **Flexible memory controller (FMC) and OCTOSPI**

  The external flash memory is accessible through NAND/NOR configurable FMC, or through OCTOSPI. The FMC also allows parallel connection of an SRAM/PSRAM instead of flash memory, if needed.

- **SD/SDIO/MMC card host interfaces (SDMMCx)**

  The SD, SDIO, *e*•MMC™ devices are interfaced through three SDMMC controllers, all connected to the STNoC interconnect. These controllers are masters of the interconnect, and originate the transfers to/from SD/SDIO/MMC memories.

Peripheral clusters

There are eight peripheral clusters in the system, briefly described hereafter:

Note: The AHBx/APBy notation means that the resources are accessed through AHBx bus, and for some of them, through AHBx bus in combination with APBy bus.

- **MLAHB150/APB1 - APB2**

  This cluster groups most of the usual MCU peripherals: timers, low-speed communication interfaces (SPI, I3C, I2C, U(S)ART, SAI), ADC, FDCAN, and MDF. Functional blocks are under control of CPU1 or CPU2, either directly or through the DMA.

- **AHB2**

  This cluster groups the ADC1/2, MDF1, IPCC1 (mailbox for interprocessor communication), DCMI, PSSI, CRC, and gives access to the three HPDMA programming ports, and to the CPU2 ICACHE and DCACHE programming ports.

- **AHB3**

  The AHB3 groups all RIF infrastructure configuration ports, and security/encryption cells (PKA, SAES, CRYP1/2, RNG1/2, HASH1/2). It is also the BKPSRAM access point.

- **AHB4/APB3**

This cluster groups important system functions as clock and power control, system configuration, most GPIO control, plus some additional timers, and low-speed communication interfaces. The BSEC is also connected to this bus, as well as some watchdogs.

- **AHB5/APB4**

  This cluster groups most of the resources control for the MPU side, including graphic and display, as well as the access to the configuration registers of the fast masters: ETH1/2, SDMMC1/2/3, USBH (EHCI/OHCI) and OTG. Additionally, the control register accesses for the DDRSS are located in the APB4. The AHB5 matrix has two independent inputs: one coming from the MCU-side (MLAHB150), and one coming from the MPU-side (STNoC).

- **AHB6/APB5**

  This cluster groups the GPIOZ control, RTC/TAMP, LPTIMERs and some low speed communication interfaces (SPI, I2C, I3C, LPUART). Functional blocks are under control of CPU1 or CPU2, either directly or through the DMA.

- **Debug**

  This cluster groups all debug features including Arm CoreSight™ infrastructure.

2.1.3 Connectivity matrix overview

The following table summarizes which master can access which the memory-mapped resource.

Table 2. Connectivity matrix

Resource	CPU1	CPU2	AXI-AP	HPDMA1/2/3	USBH EHCI USBH OHCI	OTG	ETH1/2	SDMMC1/2/3	LTDC	DCMIPP	TRACE
DDR	X	X	X	X(1)	X	X	X	X	X	X	X
Cortex-A35 debug	X	-	X	-	-	-	-	-	-	-	-
Cortex-M33 debug	-	X	X	-	-	-	-	-	-	-	-
Cortex-A35 GIC	X	-	-	-	-	-	-	-	-	-	-
FMC	X	X	X	X	-	-	-	-	-	-	-
OCTOSPI	X	X	X	X	-	-	-	-	-	-	-
STNoC registers	X	X	X	X	-	-	-	-	-	-	-
SYSRAM	X	X	X	X	X	X	X	X	X	X	X
APB debug (STM registers)	X	X	X	X	-	-	-	-	-	-	-
AHB6/APB5 peripherals	X	X	X	X	-	-	-	-	-	-	-
AHB5/APB4 peripherals	X	X	X	X	-	-	-	-	-	-	-
AHB4/APB3 peripherals	X	X	X	X	-	-	-	-	-	-	-
AHB3 peripherals	X	X	X	X	-	-	-	-	-	-	-
AHB2 peripherals	X	X	X	X	-	-	-	-	-	-	-
APB2 APB1 peripherals	X	X	X	X	-	-	-	-	-	-	-
SRAM1	X	X	X	X	X	X	X	X	X	X	X

Table 2. Connectivity matrix (continued)

Resource	CPU1	CPU2	AXI-AP	HPDMA1/2/3	USBH EHCI USBH OHCI	OTG	ETH1/2	SDMMC1/2/3	LTDC	DCMIPP	TRACE
RETRAM	X	X	X	X	X	X	X	X	X	X	X
Boot ROM	X	-	-	-	-	-	-	-	-	-	-

1. Only HPDMA1/2/3 AXI traffic

2.2 STNoC AXI network-on-chip

2.2.1 STNoC features

The STM32MP21xx STNoC AXI interconnect is built out of the ST in-house network-on-chip. Its main features are:

- 17 master ports and 8 slave ports with optimized interconnect topology from masters to slaves (no full connected)
- Source routed, packet based, wormhole switching forwarding scheme for simultaneous and parallel/pipelined accesses from master to slave
- Independent read and write input/output points to cope with AXI independent channels
- Size and frequency conversion, asynchronism management
- Fine grain automatic clock gating shutting down the clocks to unused network parts at any point in time, with zero delay clock resume
- Internal packets buffering for improved traffic efficiency
- Independent request and response network
- Software accessible registers to observe and report traffic statistics, performances, blocking sources (if any) over the network

2.2.2 STNoC interconnect configurations

The STNoC interconnect configuration is described in the figure below.

The STNoC interconnect internal router topology is described in *Figure 3*.

Note: Only the request network topology is presented, the response network has a dual topology.

Figure 2. STNoC-based MPU-side AXI interconnect

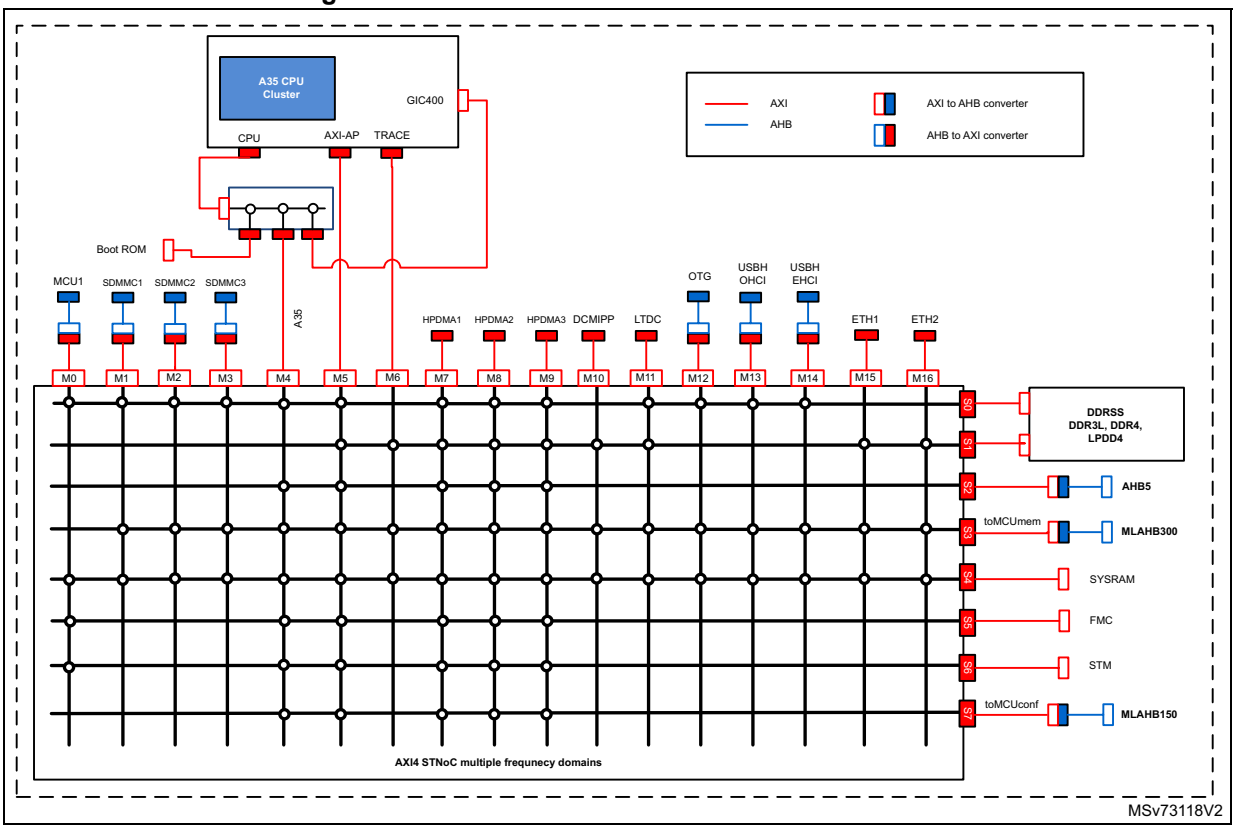
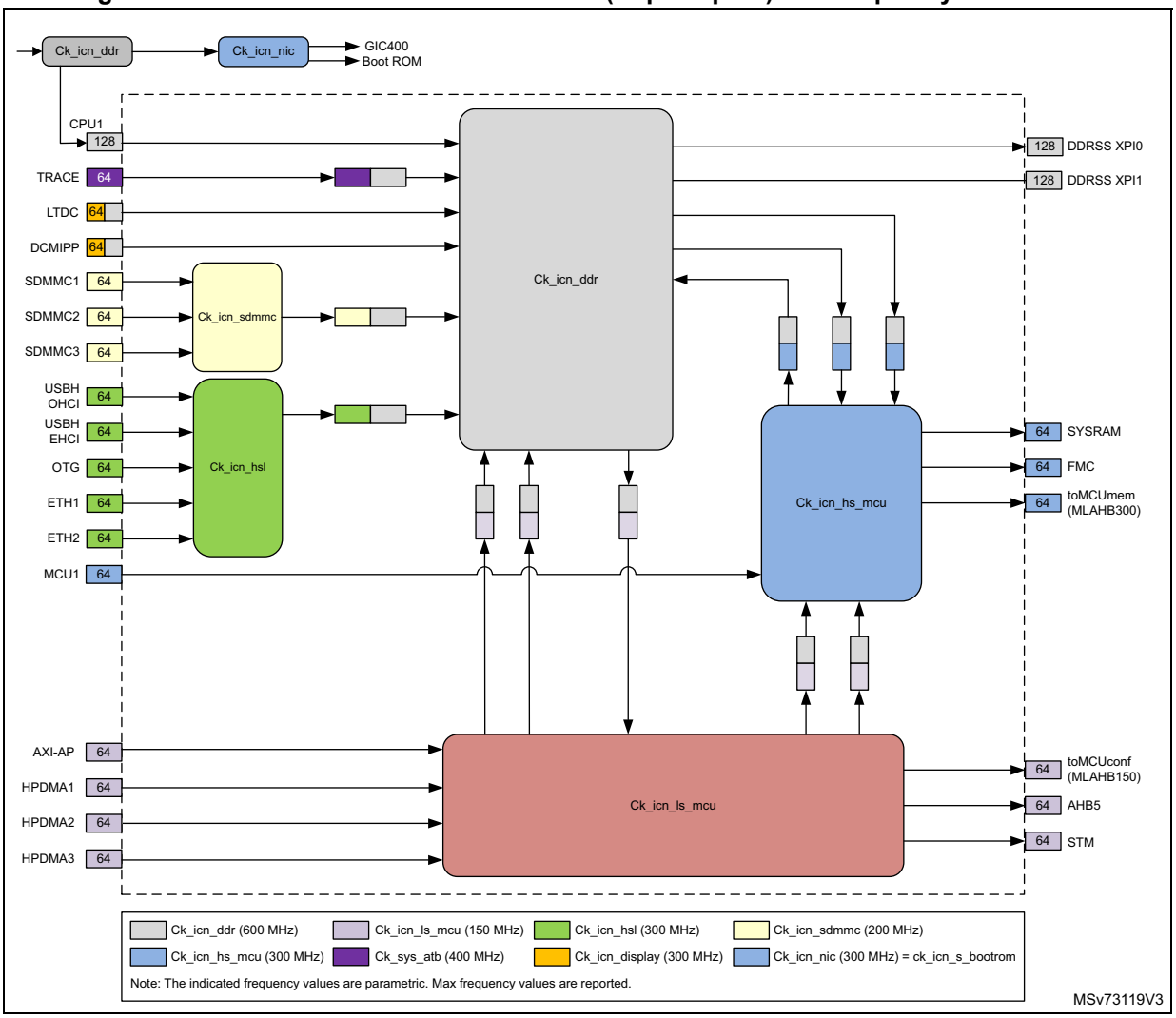


Figure 3. STNoC internal router distribution (request path) and frequency distribution



2.2.3 Master port description

The M0 to M21 master ports of AXI_STNoc are described in *Figure 2*.

- **M0**:

  M0 is used by masters from the MCU to access the memories (DDR, FMC, SYSRAM) of the MPU world and the STM. The CPU2 port reach the DDR subsystem through this port. They can only reach port 1 of the DDR subsystem. M1 is bridged to AXI externally.

- **M1/M2/M3**:

  Three AHB ports externally bridged to AXI protocol for connecting the three SDMMC controllers, with access to the SYSRAM, DDRSS XPI0 only, and MCU side memories.

- **M4**:

  M4 is the AXI access port of the CPU1 to the AXI main interconnect, 128-bit wide. The CPU subsystem is connected to DDR controller port 1 only. An intermediate NIC 400 AXI interconnect connects the CPU to the boot ROM and the GIC400.

- **M5**:

  M5 is the AXI port for the debug access port of the CPU1 cluster. It is 64-bit wide, and has access to all STNoC targets, including reaching the MCU world.

- **M6**:

  M6 is a 64-bit wide AXI port connecting to the trace output of the CPU cluster, for storing debug traces in DDR, SYSRAM. Alternatively, debug traces can also be stored on the MCU-side RAMs.

- **M7/M8/M9**:

  Three AXI 64-bit wide interfaces for connecting the AXI ports of the HPDMA, with access to all STNoC targets. The HPDMA accesses the DDR through port 1 or 2, depending if the traffic is secure or not.

- **M10**:

  M10 is a 64-bit wide AXI write-only interface for DCMIPP, with connection on the MPU-side to the DDR2 port only and SYSRAM, and access to the MCU-side memories.

- **M11**:

  M11 is a 64-bit wide AXI interface for the LTDC initiator, with connection on the MPU-side to the DDRSS XPI0 and DDRSS XPI1 (depending if traffic is secure or not) and SYSRAM, and access to the MCU-side memories.

- **M12**:

  M12 is an AHB ports externally bridged to AXI for OTG initiator, with access to the SYSRAM, DDRSS XPI0 only, and MCU side memories.

- **M13/M14**:

  These are two AHB ports externally bridged to AXI protocol for USBH OHCI and USBH EHCI, with access to the SYSRAM, DDRSS XPI0 only, and MCU side memories. M13 and M14 are not active on the interconnect at the same time.

- **M15/M16**:

  These are two 64-bit wide AXI ports for ETH1/2 with access to DDRSS XPI1 only, SYSRAM, and MCU-side memories.

2.2.4 Master port main characteristics

The following table summarizes main characteristics of the master ports.

Table 3. STNoC master port main characteristics

#	Master port	Protocol	Data bus width	Frequency (MHz)	Id width	Default QoS(1) value	Read/write issuing capability
M0	MCU1	AHB->AXI	32	300	1	15	1/1
M1	SDMMC1	AHB->AXI	32	200	1	12	1/1
M2	SDMMC2	AHB->AXI	32	200	1	12	1/1
M3	SDMMC3	AHB->AXI	32	200	1	12	1/1
M4	CPU1	AXI	128	600	6	14	16/16 (up to 16 outstanding destinations)
M5	AXI-AP	AXI	64	200	4	10	1/1
M6	TRACE	AXI	64	400	4	3	1/32 (1 single destination)
M7	HPDMA1	AXI	64	150	4	11	16/16 (up to 16 outstanding destinations)
M8	HPDMA2	AXI	64	150	4	11	16/16 (up to 16 outstanding destinations)

Table 3. STNoC master port main characteristics (continued)

#	Master port	Protocol	Data bus width	Frequency (MHz)	Id width	Default QoS(1) value	Read/write issuing capability
M9	HPDMA3	AXI	64	150	4	11	16/16 (up to 16 outstanding destinations)
M10	DCMIPP	AXI	64	300	4	8	Write only, 16 (1 single destination)
M11	LTDC	AXI	64	300	4	9	Read: 16 (up to 8 outstanding destinations) Write: 16 (single destination)
M12	OTG	AHB->AXI	32	300	1	10	1/1
M13	USBH OHCI	AHB->AXI	32	300	1	10	1/1
M14	USBH EHCI	AHB->AXI	32	300	1	10	1/1
M15	ETH1	AXI	64	300	4	7	16/16 (1 single destination)
M16	ETH2	AXI	64	300	4	7	16/16 (1 single destination)

1. The default QoS values are programmable (see *Section 23: System configuration controller (SYSCFG)*. They are used in the DDR controller when assigning a transaction to a controller internal queue. They are not used in the interconnect.

2.2.5 Slave port description

The S0 to S10 slave ports of AXI_STNoc are described in *Figure 2*.

- **S0/S1**: AXI 128-bit wide ports for accessing the DDRSS. The secure traffic, low-latency traffic and CPU1/CPU2 traffic, goes through the DDRSS XPI0. Other traffics go through the DDRSS XPI1 traffic.
- **S2**: AXI port externally bridged to AHB 32-bit wide to access the AHB5/APB4 peripheral clusters, that is the configuration peripherals for the MPU parts (such as SDMMC, DDRSS, USB, ETH) and the debug.
- **S3**: AXI port externally bridged to AHB 32-bit wide to access the MCU-side memories. This port leads to an input of the MLAHB300 AHB matrix.
- **S4**: AXI 64-bit wide port to access the SYSRAM
- **S5**: AXI 64-bit wide port for accessing the FMC target device
- **S6**: AXI 64-bit wide target port for accessing the CPU1 STM port
- **S7**: AXI port externally bridged to AHB 32-bit wide to access the MCU-side peripherals. This port leads to an input of the MLAHB150 AHB matrix

2.2.6 Slave port main characteristics

The following table lists main characteristics of the slave ports.

Table 4. Slave port main characteristics

Slave port	Port number	Protocol	Data bus width	Frequency (MHz)
DDRSS XPI0	S0	AXI	128	600
DDRSS XPI1	S1	AXI	128	600
toMCUmem	S2	AXI->AHB	32	300
MPU_AHB5	S3	AXI->AHB	32	150
SYSRAM	S4	AXI	64	400
FMC	S5	AXI	64	400
STM	S6	AXI	64	200
toMCUconf	S7	AXI->AHB	32	150

2.3 Multi-layer AHB interconnect

2.3.1 Multi-layer AHB features

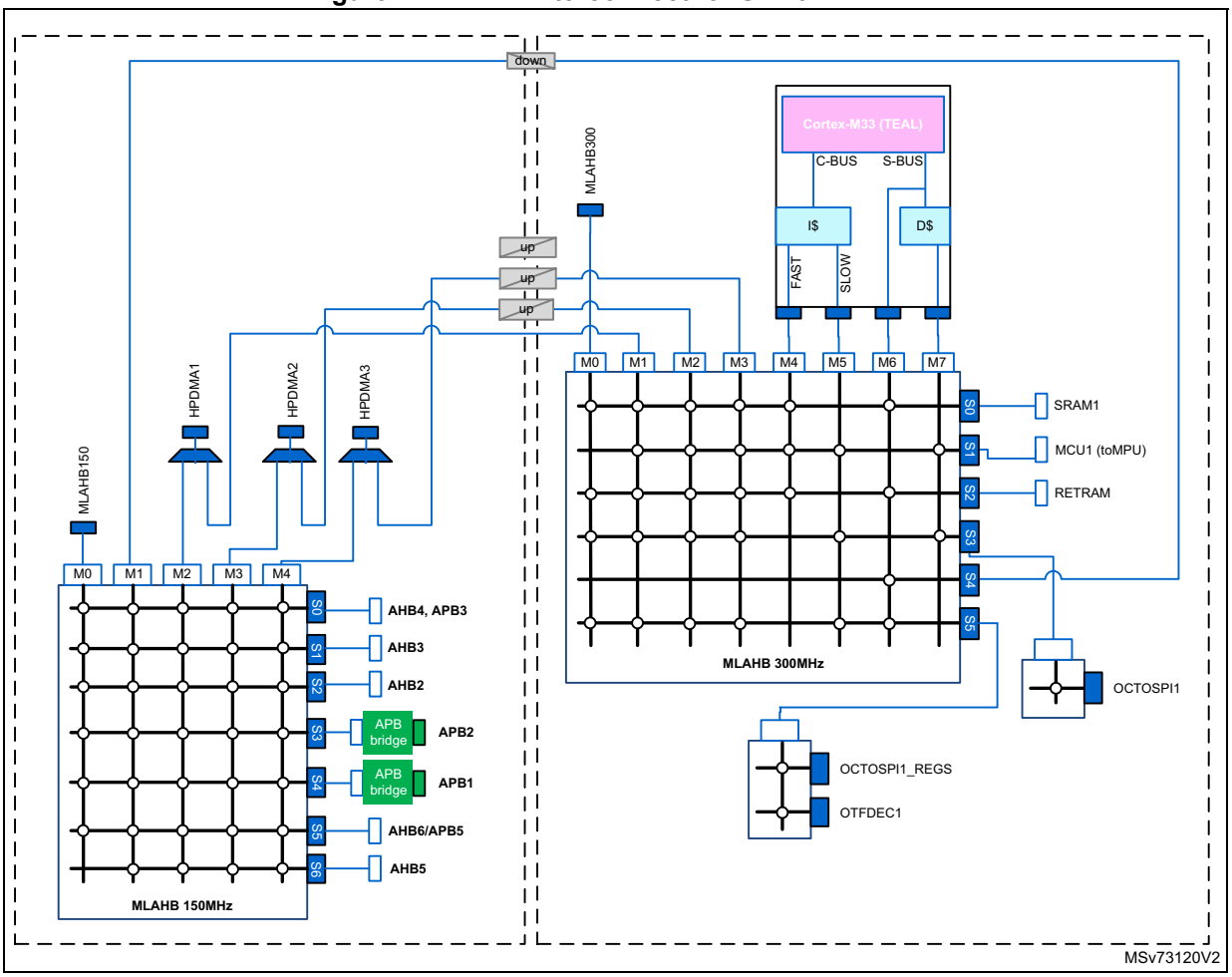
The multi-layer AHB main features are:

- Spread in two matrices working at different frequencies to trade off high performances versus low-latency accesses to peripherals
- MLAHB300 connects eight initiator ports to six target ports, running at 300 MHz, and allowing mainly accesses from the CPU2 and HPDMA to the MCU-side memories.
- MLAHB150 connects five initiator ports to seven target ports, running at 150 MHz, and allowing mainly access to the CPU2 and HPDMA to the MCU side peripherals.
- Simultaneous accesses support from different masters to the slaves connected to separated layers

2.3.2 Multi-layer AHB interconnect configuration

The multi-layer AHB interconnect configuration is described in the figure below.

Figure 4. MLAHB interconnect for STM32MP21xx



2.3.3 Multi-layer AHB 300 MHz master port characteristics

The M0 to M7 slave ports of MLAHB300 are described in *Figure 4*.

- **M0**:
  M0 is connected to an MPU interconnect master port, allowing masters from the MPU subsystem to access the memories located in the MCU-side.
- **M1/M2/M3**:
  M1/M2/M3 ports connecting the demultiplexed AHB ports of the HPDMA, to give access to the HPDMA AHB ports to the AHB memories.
- **M4**:
  CPU2 I$ (C-BUS) fast access, dedicated to embedded memories with low-access latencies
- **M5**:
  CPU2 I$ (C-BUS) slow access, dedicated to memories with high-access latencies, namely the DDR on the MPU side or the OCTOSPI connected memory device
- **M6**:
  CPU2 D$ (S-BUS) bypass access, dedicated to embedded memories with low access latencies. The traffic from the Cortex-M33 towards SYSRAM (ranges above 0x2000 0000), SRAM1, RETRAM and peripherals is routed through this port.
- **M7**:
  CPU2 D$ (S-BUS) access, dedicated to high-latency memories such as DDR on the MPU side, or OCTOSPI connected device. The traffic from the Cortex-M33 towards FMC, DDR, and OCTOSPI memories is routed through this port.

The following table lists the main characteristics of MLAHB300 master ports.

Table 5. MLAHB300 master port main characteristics

Master port	Port number	Protocol	Data bus width (bits)	Synchronicity
MLAHB300 (toMCUMem)	M0	AHB	32	Sync
HPDMA1	M1			
HPDMA2	M2			
HPDMA3	M3			
Cortex-M33_i$_fast	M4			
Cortex-M33_i$_slow	M5			
Cortex-M33_D$_bypass	M6			
Cortex-M33_D$	M7			

2.3.4 Multi-layer AHB 300 MHz slave port characteristics

The S0 to S6 slave ports of MLAHB300 are described in *Figure 4*.

- **S0**: AHB ports to SRAM1
- **S1**: AHB port to access the MPU-side memories. When directed towards DDR, all traffics coming from this port are routed to DDRSS XPI0 (secure-enabled one), including HPDMA traffic from their AHB port.
- **S2**: AHB port to access the retention RAM
- **S3**: AHB port to access the OCTOSPI cluster

- **S4**: AHB port bridging to the MLAHB150 for accessing the peripherals on MCU side. The HPDMA has a dedicated access on the MLAHB150 and must not use this port for accessing the MCU-side peripherals. There is a downsync block on this path towards the MLAHB150.
- **S5**: AHB port accessing configuration registers for the OTFDEC1, and OCTOSPI1/2 registers.

The following table lists the main characteristics of the MLAHB300 target ports.

Table 6. MLAHB300 target port main characteristics

Slave port	Port number	Protocol	Data bus width	Synchronicity
SRAM1	S0	AHB	32	Sync
MCU1 (toMPU)	S1			
RETRAM	S2			
OCTOSPI	S3			
ToMLAHB150	S4			
OCTO_CONFIG	S5			

2.3.5 Multi-layer AHB 150 MHz master port characteristics

The M0 to M4 master ports of MLAHB150 are described in *Figure 4*.

- **M0**: AHB port receiving traffic towards the MCU-side peripherals from the MPU initiators
- **M1**: AHB port receiving traffic from the MLAHB300 towards the MCU peripherals. Only the CPU2 initiated traffic must go through this port, since the HPDMA AHB ports have a dedicated access port on this matrix.
- **M2/M3/M4**: AHB port receiving traffic from the demultiplexed HPDMA1/2/3 AHB ports aimed at peripherals

The following table lists the main characteristics of the MLAHB150 master ports.

Table 7. MLAHB150 master port main characteristics.

Master port	Port number	Protocol	Data bus width	Synchronicity
MLAHB150 (toMCUconf)	M0	AHB	32	Sync
fromMLAHB300	M1			
HPDMA1	M2			
HPDMA2	M3			
HPDMA3	M4			

2.3.6 Multi-layer AHB 150 MHz slave port characteristics

The S0 to S8 slave ports of MLAHB150 are described in *Figure 4*.

- **S0**: AHB target port to connect to the AHB4/APB3 peripherals, where mainly GPIOs are accessible
- **S1**: AHB target port to connect to AHB3 peripherals, where mainly security peripherals are located
- **S2**: AHB target port to connect to the AHB2 peripherals, where mainly registers for configuring different cells (HPDMA, CPU2 cache) and ADC are located

- **S3**: AHB target port to connect to the APB2 peripherals through AHB2APB in-house bridge
- **S4**: AHB target port to connect to the APB1 peripherals through AHB2APB in-house bridge
- **S5**: AHB target port to connect to AHB6/APB5 peripherals
- **S6**: AHB target port to connect to AHB5 peripherals, where the resources control for the MPU side are located

The table below lists the main characteristics of the MLAHB150 target ports.

Table 8. MLAHB150 slave ports main characteristics

Slave port	Port number	Protocol	Data bus width	Synchronicity
toAHB4	S0	AHB	32	Sync
toAHB3	S1			
toAHB2	S2			
toAPB2	S3			
toAPB1	S4			
toAHB6	S5			
toAHB5	S6			

2.4 Memory organization

2.4.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

The access to some part of the address space is master dependent, see *Section 2.4.2: Memory map and register boundary addresses* for details.

2.4.2 Memory map and register boundary addresses

Figure 5. Memory map

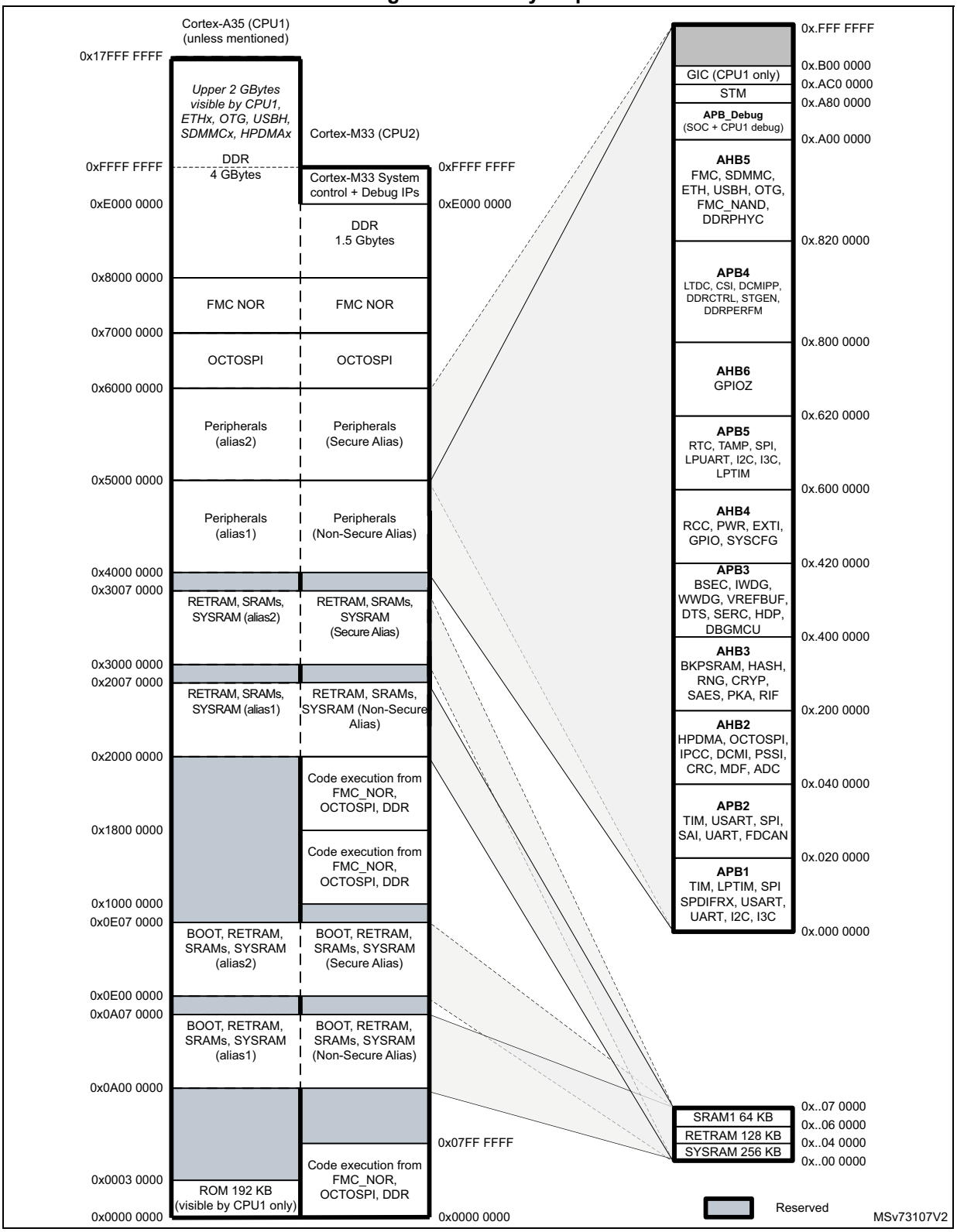


Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
DDR access area for CPU1, private periphs for CPU2	0xE0100000 - 0xFFFFFFFF	523264	Reserved	-
	0xE00FF000 - 0xE00FFFFF	4	ROM table	*Section 80: Debug support (DBG)*
	0xE0044000 - 0xE00FEFFF	748	Reserved	-
	0xE0043000 - 0xE0043FFF	4	Reserved (MTB)	-
	0xE0042000 - 0xE0042FFF	4	CTI	*Section 80: Debug support (DBG)*
	0xE0041000 - 0xE0041FFF	4	ETM	
	0xE0040000 - 0xE0040FFF	4	TPIU	
	0xE002F000 - 0xE003FFFF	68	Reserved	-
	0xE002E000 - 0xE002EFFF	4	SCS non-secure alias	*Section 80: Debug support (DBG)*
	0xE000F000 - 0xE002DFFF	124	Reserved	-
	0xE000E000 - 0xE000EFFF	4	SCS	*Section 80: Debug support (DBG)*
	0xE0003000 - 0xE000DFFF	44	Reserved	-
	0xE0002000 - 0xE0002FFF	4	BPU	*Section 80: Debug support (DBG)*
	0xE0001000 - 0xE0001FFF	4	DWT	
	0xE0000000 - 0xE0000FFF	4	ITM	
AXIM	0x5AC00000 - 0x5AFFFFFF	4096	Reserved	-
	0x5A800000 - 0x5ABFFFFF	4096	STM	*Section 80: Debug support (DBG)*
APB_Debug	0x5A0B0000 - 0x5A7FFFFF	7424	Reserved	-
	0x5A0A0000 - 0x5A0AFFFF	64	SWO	*Section 80: Debug support (DBG)*
	0x5A090000 - 0x5A09FFFF	64	CTI1	
	0x5A080000 - 0x5A08FFFF	64	CTI0	
	0x5A070000 - 0x5A07FFFF	64	STM registers	
	0x5A060000 - 0x5A06FFFF	64	TSGEN1	
	0x5A050000 - 0x5A05FFFF	64	REPLICATOR	
	0x5A040000 - 0x5A04FFFF	64	TPIU	
	0x5A030000 - 0x5A03FFFF	64	ETR	
	0x5A020000 - 0x5A02FFFF	64	ETF	
	0x5A010000 - 0x5A01FFFF	64	Funnel	*Section 80: Debug support (DBG)*
	0x5A000000 - 0x5A00FFFF	64	SoC ROM table	
AHB5	0x59000000 - 0x59FFFFFF	16384	Reserved	-
	0x58C00000 - 0x58FFFFFF	4096	DDRPHYC	*Section 14.5.194: DDRPHYC register map*
	0x58811000 - 0x58BFFFFF	4028	Reserved	-
	0x58810000 - 0x58810FFF	4	FMC_NAND	External memory access
	0x58804000 - 0x5880FFFF	48	Reserved	-
	0x58800000 - 0x58803FFF	16	CA35SS	*Section 3.6.5: CA35SS SSC register map*
	0x58340000 - 0x587FFFFF	4864	Reserved	-
	0x58300000 - 0x5833FFFF	256	OTG	*Section 74.14.63: OTG register map*
	0x582F0400 - 0x582FFFFF	63	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB5 (cont'd)	0x582F0000 - 0x582F03FF	1	USBH_EHCI	*Section 75.5.12: USBH register map*
	0x582E0400 - 0x582EFFFF	63	Reserved	-
	0x582E0000 - 0x582E03FF	1	USBH_OHCI	*Section 75.5.12: USBH register map*
	0x582D4000 - 0x582DFFFF	48	Reserved	-
	0x582D0000 - 0x582D3FFF	16	ETH2	*Section 77.11.1: Ethernet register maps*
	0x582C4000 - 0x582CFFFF	48	Reserved	-
	0x582C0000 - 0x582C3FFF	16	ETH1	*Section 77.11.1: Ethernet register maps*
	0x58240400 - 0x582BFFFF	511	Reserved	-
	0x58240000 - 0x582403FF	1	SDMMC3	*Section 34.10.26: SDMMC register map*
	0x58230400 - 0x5823FFFF	63	Reserved	-
	0x58230000 - 0x582303FF	1	SDMMC2	*Section 34.10.26: SDMMC register map*
	0x58220400 - 0x5822FFFF	63	Reserved	-
	0x58220000 - 0x582203FF	1	SDMMC1	*Section 34.10.26: SDMMC register map*
	0x58200400 - 0x5821FFFF	127	Reserved	-
	0x58200000 - 0x582003FF	1	FMC	*Section 32.12: FMC register map*
APB4	0x580C0400 - 0x581FFFFF	1279	Reserved	-
	0x580C0000 - 0x580C03FF	1	DDRPERFM	*Section 15.5.25: DDRPERFM register map*
	0x58081000 - 0x580BFFFF	252	Reserved	-
	0x58080000 - 0x58080FFF	4	STGENC	*Section 59.4.32: STGEN register map*
	0x58071000 - 0x5807FFFF	60	Reserved	-
	0x58070000 - 0x58070FFF	4	STGENR	*Section 59.4.32: STGEN register map*
	0x58060000 - 0x5806FFFF	64	Reserved	-
	0x58050000 - 0x5805FFFF	64	DDRDBG	*Section 12.6.17: DDRSS register map*
	0x58040000 - 0x5804FFFF	64	DDRCTRL	*Section 13.7.102: DDRCTRL register map*
	0x58031000 - 0x5803FFFF	60	Reserved	-
	0x58030000 - 0x58030FFF	4	DCMIPP	*Section 42.15: DCMIPP register map*
	0x58022000 - 0x5802FFFF	56	Reserved	-
	0x58020000 - 0x58021FFF	8	CSI	*Section 43.10.11: CSI-2 Host and PHY register map*
	0x58010400 - 0x5801FFFF	63	Reserved	-
	0x58010000 - 0x580103FF	1	LTDC	*Section 46.8.58: LTDC register map*
	0x58000000 - 0x5800FFFF	64	Reserved	-
AHB6	0x56200400 - 0x57FFFFFF	30718	Reserved	-
	0x56200000 - 0x562003FF	1	GPIOZ	*Section 22.4.35: GPIO register map*
APB5	0x56080400 - 0x561FFFFF	1534	Reserved	-
	0x56080000 - 0x560803FF	1	I3C3	*Section 66.16.34: I3C register map*
	0x56070400 - 0x5607FFFF	63	Reserved	-
	0x56070000 - 0x560703FF	1	LPTIM5	*Section 58.7.25: LPTIM register map*
	0x56060400 - 0x5606FFFF	63	Reserved	-
	0x56060000 - 0x560603FF	1	LPTIM4	*Section 58.7.25: LPTIM register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)[1] (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB5 (cont'd)	0x56050400 - 0x5605FFFF	63	Reserved	-
	0x56050000 - 0x560503FF	1	LPTIM3	*Section 58.7.25: LPTIM register map*
	0x56040400 - 0x5604FFFF	63	Reserved	-
	0x56040000 - 0x560403FF	1	I2C3	*Section 65.9.17: I2C register map*
	0x56030400 - 0x5603FFFF	63	Reserved	-
	0x56030000 - 0x560303FF	1	LPUART1	*Section 68.7.20: LPUART register map*
	0x56020400 - 0x5602FFFF	63	Reserved	-
	0x56020000 - 0x560203FF	1	SPI6	*Section 69.11.21: SPI/I2S register map*
	0x56010400 - 0x5601FFFF	63	Reserved	-
	0x56010000 - 0x560103FF	1	TAMP	*Section 64.6.30: TAMP register map*
	0x56000400 - 0x5600FFFF	63	Reserved	-
	0x56000000 - 0x560003FF	1	RTC	*Section 63.6.33: RTC register map*
AHB4	0x542D0400 - 0x55FFFFFF	29887	Reserved	-
	0x542D0000 - 0x542D03FF	1	EXTI2	*Section 29.7.86: EXTI2 register map*
	0x542C0400 - 0x542CFFFF	63	Reserved	-
	0x542C0000 - 0x542C03FF	1	GPIOI	*Section 22.4.35: GPIO register map*
	0x542B0400 - 0x542BFFFF	63	Reserved	-
	0x542B0000 - 0x542B03FF	1	GPIOH	*Section 22.4.35: GPIO register map*
	0x542A0400 - 0x542AFFFF	63	Reserved	-
	0x542A0000 - 0x542A03FF	1	GPIOG	*Section 22.4.35: GPIO register map*
	0x54290400 - 0x5429FFFF	63	Reserved	-
	0x54290000 - 0x542903FF	1	GPIOF	*Section 22.4.35: GPIO register map*
	0x54280400 - 0x5428FFFF	63	Reserved	-
	0x54280000 - 0x542803FF	1	GPIOE	*Section 22.4.35: GPIO register map*
	0x54270400 - 0x5427FFFF	63	Reserved	-
	0x54270000 - 0x542703FF	1	GPIOD	*Section 22.4.35: GPIO register map*
	0x54260400 - 0x5426FFFF	63	Reserved	-
	0x54260000 - 0x542603FF	1	GPIOC	*Section 22.4.35: GPIO register map*
	0x54250400 - 0x5425FFFF	63	Reserved	-
	0x54250000 - 0x542503FF	1	GPIOB	*Section 22.4.35: GPIO register map*
	0x54240400 - 0x5424FFFF	63	Reserved	-
	0x54240000 - 0x542403FF	1	GPIOA	*Section 22.4.35: GPIO register map*
	0x54230000 - 0x5423FFFF	64	SYSCFG	*Section 23.2.58: SYSCFG register map*
	0x54220400 - 0x5422FFFF	63	Reserved	-
	0x54220000 - 0x542203FF	1	EXTI1	*Section 29.7.85: EXTI1 register map*
	0x54210400 - 0x5421FFFF	63	Reserved	-
	0x54210000 - 0x542103FF	1	PWR	*Section 19.12.27: PWR register map*
	0x54200000 - 0x5420FFFF	64	RCC	*Section 20.8.206: RCC register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB3	0x540A0400 - 0x541FFFFF	1407	Reserved	
	0x540A0000 - 0x540A03FF	1	DBGMCU	*Section : DBGMCU register map*
	0x54090400 - 0x5409FFFF	3	Reserved	-
	0x54090000 - 0x540903FF	1	HDP	*Section 78.3.10: HDP register map*
	0x54081000 - 0x5408FFFF	60	Reserved	-
	0x54080000 - 0x54080FFF	4	SERC	*Section 79.6.24: SERC register map*
	0x54071000 - 0x5407FFFF	60	Reserved	-
	0x54070000 - 0x54070FFF	4	DTS	*Section 37.5.5: DTS register map*
	0x54060400 - 0x5406FFFF	63	Reserved	-
	0x54060000 - 0x540603FF	1	VREFBUF	*Section 38.6.3: VREFBUF register map*
	0x54050400 - 0x5405FFFF	63	Reserved	-
	0x54050000 - 0x540503FF	1	WWDG1	*Section 62.6.8: WWDG register map*
	0x54040400 - 0x5404FFFF	63	Reserved	-
	0x54040000 - 0x540403FF	1	IWDG4	*Section 61.7.12: IWDG register map*
	0x54030400 - 0x5403FFFF	63	Reserved	-
	0x54030000 - 0x540303FF	1	IWDG3	*Section 61.7.12: IWDG register map*
	0x54020400 - 0x5402FFFF	63	Reserved	-
	0x54020000 - 0x540203FF	1	IWDG2	*Section 61.7.12: IWDG register map*
	0x54010400 - 0x5401FFFF	63	Reserved	-
	0x54010000 - 0x540103FF	1	IWDG1	*Section 61.7.12: IWDG register map*
	0x54001000 - 0x5400FFFF	60	Reserved	-
	0x54000000 - 0x54000FFF	4	BSEC	*Section 6.5.34: BSEC register map*
AHB3	0x52131000 - 0x53FFFFFF	31548	Reserved	-
	0x52130000 - 0x52130FFF	4	RISAB5	*Section 9.5.20: RISAB register map*
	0x52111000 - 0x5212FFFF	124	Reserved	-
	0x52110000 - 0x52110FFF	4	RISAB3	*Section 9.5.20: RISAB register map*
	0x52101000 - 0x5210FFFF	60	Reserved	-
	0x52100000 - 0x52100FFF	4	RISAB2	*Section 9.5.20: RISAB register map*
	0x520F1000 - 0x520FFFFF	60	Reserved	-
	0x520F0000 - 0x520F0FFF	4	RISAB1	*Section 9.5.20: RISAB register map*
	0x520D2000 - 0x520EFFFF	119	Reserved	-
	0x520D0000 - 0x520D1FFF	8	RISAF4	*Section 10.5.27: RISAF register map*
	0x520B1000 - 0x520CFFFF	124	Reserved	-
	0x520B0000 - 0x520B0FFF	4	RISAF2	*Section 10.5.27: RISAF register map*
	0x520A1000 - 0x520AFFFF	60	Reserved	-
	0x520A0000 - 0x520A0FFF	4	RISAF1	*Section 10.5.27: RISAF register map*
	0x52090400 - 0x5209FFFF	63	Reserved	-
	0x52090000 - 0x520903FF	1	IAC	*Section 11.6.10: IAC register map*
	0x52081000 - 0x5208FFFF	60	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB (cont'd)	0x52080000 - 0x52080FFF	4	RIFSC	*Section 8.5.17: RIFSC register map*
	0x52071000 - 0x5207FFFF	60	Reserved	-
	0x52070000 - 0x52070FFF	4	RAMCFG	*Section 16.6.28: RAMCFG register map*
	0x52041000 - 0x5206FFFF	188	Reserved	-
	0x52040000 - 0x52040FFF	4	CRYP2	*Section 49.8.27: CRYP register map*
	0x52039C00 - 0x5203FFFF	25	Reserved	-
	0x52037C00 - 0x52039BFF	8	Reserved	-
	0x52034000 - 0x52037BFF	15	Reserved	-
	0x52032000 - 0x52033FFF	8	PKA	*Section 52.7.9: PKA register map*
	0x52031000 - 0x52031FFF	4	Reserved	-
	0x52030C00 - 0x52030FFF	1	SAES	*Section 48.8.25: SAES register map*
	0x52030800 - 0x52030BFF	1	RNG1	*Section 47.7.10: RNG register map*
	0x52030400 - 0x520307FF	1	HASH1	*Section 50.6.12: HASH register map*
	0x52030000 - 0x520303FF	1	CRYP1	*Section 49.8.27: CRYP register map*
	0x52020400 - 0x5202FFFF	63	Reserved	-
	0x52020000 - 0x520203FF	1	RNG2	*Section 47.7.10: RNG register map*
	0x52010400 - 0x5201FFFF	63	Reserved	-
	0x52010000 - 0x520103FF	1	HASH2	*Section 50.6.12: HASH register map*
	0x52002000 - 0x5200FFFF	56	Reserved	-
	0x52000000 - 0x52001FFF	8	BKPSRAM	Memory access
AHB2	0x505F0400 - 0x51FFFFFF	27711	Reserved	-
	0x504F0300 - 0x505F03FF	0.25	ADCC2	*Section 36.8: ADC register map*
	0x504F0100 - 0x504F02FF	0.5	Reserved	-
	0x504F0000 - 0x504F00FF	0.25	ADC2	*Section 36.8: ADC register map*
	0x504E0400 - 0x504EFFFF	63	Reserved	-
	0x504E0300 - 0x505E03FF	0.25	ADCC1	*Section 36.8: ADC register map*
	0x504E0100 - 0x504E02FF	0.5	Reserved	-
	0x504E0000 - 0x504E00FF	0.25	ADC1	*Section 36.8: ADC register map*
	0x504D1000 - 0x504DFFFF	60	Reserved	-
	0x504D0000 - 0x504D0FFF	4	MDF1	*Section 39.8.25: MDF register map*
	0x504C0400 - 0x504CFFFF	63	Reserved	-
	0x504C0000 - 0x504C03FF	1	CRC	*Section 30.4.6: CRC register map*
	0x504B0400 - 0x504BFFFF	63	Reserved	-
	0x504B0000 - 0x504B03FF	1	PSSI	*Section 44.5.12: PSSI register map*
	0x504A0400 - 0x504AFFFF	63	Reserved	-
	0x504A0000 - 0x504A03FF	1	DCMI	*Section 41.5.16: DCMI register map*
	0x50490400 - 0x5049FFFF	63	Reserved	-
	0x50490000 - 0x504903FF	1	IPCC1	*Section 21.4.19: IPCC register map*
	0x50480400 - 0x5048FFFF	63	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB2 (cont'd)	0x50480000 - 0x504803FF	1	DCACHE	*Section 18.7.15: DCACHE register map*
	0x50470400 - 0x5047FFFF	63	Reserved	-
	0x50470000 - 0x504703FF	1	ICACHE	*Section 17.7.12: ICACHE register map*
	0x50460400 - 0x5046FFFF	63	Reserved	-
	0x50450400 - 0x504603FF	64	Reserved	-
	0x50450000 - 0x504503FF	1	OTFDEC1	*Section 51.6.18: OTFDEC register map*
	0x50430400 - 0x504403FF	64	Reserved	-
	0x50430000 - 0x504303FF	1	OCTOSPI1	*Section 33.7.33: OCTOSPI register map*
	0x50421000 - 0x5042FFFF	60	Reserved	-
	0x50420000 - 0x50420FFF	4	HPDMA3	*Section 25.8.40: HPDMA register map*
	0x50411000 - 0x5041FFFF	60	Reserved	-
	0x50410000 - 0x50410FFF	4	HPDMA2	*Section 25.8.40: HPDMA register map*
	0x50401000 - 0x5040FFFF	60	Reserved	-
	0x50400000 - 0x50400FFF	4	HPDMA1	*Section 25.8.40: HPDMA register map*
APB2	0x50370400 - 0x503FFFFF	575	Reserved	-
	0x50370000 - 0x503703FF	1	UART7	*Section 67.8.22: USART register map*
	0x50340400 - 0x5036FFFF	191	Reserved	-
	0x50340000 - 0x503403FF	1	SAI4	*Section 70.6.24: SAI register map*
	0x50330400 - 0x5033FFFF	63	Reserved	-
	0x50330000 - 0x503303FF	1	USART1	*Section 67.8.22: USART register map*
	0x50312800 - 0x5032FFFF	108	Reserved	-
	0x50310000 - 0x503127FF	10	FDCANRAM	Memory access
	0x50300400 - 0x5030FFFF	63	Reserved	-
	0x50300000 - 0x503003FF	1	CCU	*Section 72.7.7: CCU register map*
	0x502E0400 - 0x502FFFFF	127	Reserved	-
	0x502E0000 - 0x502E03FF	1	FDCAN2	*Section 72.5.47: FDCAN register map*
	0x502D0400 - 0x502DFFFF	63	Reserved	-
	0x502D0000 - 0x502D03FF	1	FDCAN1	*Section 72.5.47: FDCAN register map*
	0x502B0400 - 0x502CFFFF	127	Reserved	-
	0x502B0000 - 0x502B03FF	1	SAI3	*Section 70.6.24: SAI register map*
	0x502A0400 - 0x502AFFFF	63	Reserved	-
	0x502A0000 - 0x502A03FF	1	SAI2	*Section 70.6.24: SAI register map*
	0x50290400 - 0x5029FFFF	63	Reserved	-
	0x50290000 - 0x502903FF	1	SAI1	*Section 70.6.24: SAI register map*
	0x50280400 - 0x5028FFFF	63	Reserved	-
	0x50280000 - 0x502803FF	1	SPI5	*Section 69.11.21: SPI/I2S register map*
	0x50270400 - 0x5027FFFF	63	Reserved	-
	0x50270000 - 0x502703FF	1	TIM17	*Section 57.8.26: TIM16/TIM17 register map*
	0x50260400 - 0x5026FFFF	63	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB2 (cont'd)	0x50260000 - 0x502603FF	1	TIM16	*Section 57.8.26: TIM16/TIM17 register map*
	0x50250400 - 0x5025FFFF	63	Reserved	-
	0x50250000 - 0x502503FF	1	TIM15	*Section 54.5.30: TIMx register map*
	0x50240400 - 0x5024FFFF	63	Reserved	-
	0x50240000 - 0x502403FF	1	SPI4	*Section 69.11.21: SPI/I2S register map*
	0x50230400 - 0x5023FFFF	63	Reserved	-
	0x50230000 - 0x502303FF	1	SPI1	*Section 69.11.21: SPI/I2S register map*
	0x50220400 - 0x5022FFFF	63	Reserved	-
	0x50220000 - 0x502203FF	1	USART6	*Section 67.8.22: USART register map*
	0x50210400 - 0x5021FFFF	63	Reserved	-
	0x50210000 - 0x502103FF	1	TIM8	*Section 53.6.36: TIMx register map*
	0x50200400 - 0x5020FFFF	63	Reserved	-
	0x50200000 - 0x502003FF	1	TIM1	*Section 53.6.36: TIMx register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)[1] (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB1	0x501D0400 - 0x501FFFFF	191	Reserved	-
	0x501D0000 - 0x501D03FF	1	TIM11	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x501C0400 - 0x501CFFFF	63	Reserved	-
	0x501C0000 - 0x501C03FF	1	TIM10	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x501A0400 - 0x501BFFFF	127	Reserved	-
	0x501A0000 - 0x501A03FF	1	I3C2	*Section 66.16.34: I3C register map*
	0x50190400 - 0x5019FFFF	63	Reserved	-
	0x50190000 - 0x501903FF	1	I3C1	*Section 66.16.34: I3C register map*
	0x50180400 - 0x5018FFFF	63	Reserved	-
	0x50180000 - 0x501803FF	1	I2C2	*Section 65.9.17: I2C register map*
	0x50170400 - 0x5017FFFF	63	Reserved	-
	0x50170000 - 0x501703FF	1	I2C1	*Section 65.9.17: I2C register map*
	0x50110400 - 0x5016FFFF	383	Reserved	-
	0x50110000 - 0x501103FF	1	UART5	*Section 67.8.22: USART register map*
	0x50100400 - 0x5010FFFF	63	Reserved	-
	0x50100000 - 0x501003FF	1	UART4	*Section 67.8.22: USART register map*
	0x500F0400 - 0x500FFFFF	63	Reserved	-
	0x500F0000 - 0x500F03FF	1	USART3	*Section 67.8.22: USART register map*
	0x500E0400 - 0x500EFFFF	63	Reserved	-
	0x500E0000 - 0x500E03FF	1	USART2	*Section 67.8.22: USART register map*
	0x500D0400 - 0x500DFFFF	63	Reserved	-
	0x500D0000 - 0x500D03FF	1	SPDIFRX	*Section 71.5.13: SPDIFRX interface register map*
	0x500C0400 - 0x500CFFFF	63	Reserved	-
	0x500C0000 - 0x500C03FF	1	SPI3	*Section 69.11.21: SPI/I2S register map*
	0x500B0400 - 0x500BFFFF	63	Reserved	-
	0x500B0000 - 0x500B03FF	1	SPI2	*Section 69.11.21: SPI/I2S register map*
	0x500A0400 - 0x500AFFFF	63	Reserved	-
	0x500A0000 - 0x500A03FF	1	LPTIM2	*Section 58.7.25: LPTIM register map*
	0x50090400 - 0x5009FFFF	63	Reserved	-
	0x50090000 - 0x500903FF	1	LPTIM1	*Section 58.7.25: LPTIM register map*
	0x50080400 - 0x5008FFFF	63	Reserved	-
	0x50080000 - 0x500803FF	1	TIM14	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x50070400 - 0x5007FFFF	63	Reserved	-
	0x50070000 - 0x500703FF	1	TIM13	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x50060400 - 0x5006FFFF	63	Reserved	-
	0x50060000 - 0x500603FF	1	TIM12	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB1 (cont'd)	0x50050400 - 0x5005FFFF	63	Reserved	-
	0x50050000 - 0x500503FF	1	TIM7	*Section 55.4.14: TIMx register map*
	0x50040400 - 0x5004FFFF	63	Reserved	-
	0x50040000 - 0x500403FF	1	TIM6	*Section 55.4.14: TIMx register map*
	0x50030400 - 0x5003FFFF	63	Reserved	-
	0x50030000 - 0x500303FF	1	TIM5	*Section 54.5.30: TIMx register map*
	0x50020400 - 0x5002FFFF	63	Reserved	-
	0x50020000 - 0x500203FF	1	TIM4	*Section 54.5.30: TIMx register map*
	0x50010400 - 0x5001FFFF	63	Reserved	-
	0x50010000 - 0x500103FF	1	TIM3	*Section 54.5.30: TIMx register map*
	0x50000400 - 0x5000FFFF	63	Reserved	-
	0x50000000 - 0x500003FF	1	TIM2	*Section 54.5.30: TIMx register map*
	0x4AC62000 - 0x4FFFFFFF	85624	Reserved	-
	0x4AC60000 - 0x4AC61FFF	8	GICV	*Section 28.8.15: GICV register map*
	0x4AC41000 - 0x4AC5FFFF	124	Reserved	-
	0x4AC40000 - 0x4AC40FFF	4	GICH	*Section 28.7.9: GICH register map*
	0x4AC20000 - 0x4AC3FFFF	128	GICC	*Section 28.6.18: GICC register map*
	0x4AC11000 - 0x4AC1FFFF	60	Reserved	-
	0x4AC10000 - 0x4AC10FFF	4	GICD	*Section 28.5.35: GICD register map*
	0x4AC00000 - 0x4AC0FFFF	64	Reserved	-
	0x4A800000 - 0x4ABFFFFF	4096	STM	

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB_Debug	0x4A350000 - 0x4A7FFFFF	4800	Reserved	-
	0x4A340000 - 0x4A34FFFF	64	CA35_ETM1	*Section 80: Debug support (DBG)*
	0x4A330000 - 0x4A33FFFF	64	CA35_PMU1	
	0x4A320000 - 0x4A32FFFF	64	CA35_CTI1	
	0x4A310000 - 0x4A31FFFF	64	CA35_DBG1	
	0x4A250000 - 0x4A30FFFF	768	Reserved	-
	0x4A240000 - 0x4A24FFFF	64	CA35_ETM0	*Section 80: Debug support (DBG)*
	0x4A230000 - 0x4A23FFFF	64	CA35_PMU0	
	0x4A220000 - 0x4A22FFFF	64	CA35_CTI0	
	0x4A210000 - 0x4A21FFFF	64	CA35_DBG0	
	0x4A200000 - 0x4A20FFFF	64	CA35SS ROM table	
	0x4A0C0000 - 0x4A1FFFFF	1280	Reserved	-
	0x4A0B0000 - 0x4A0BFFFF	64	Reserved	-
	0x4A0A0000 - 0x4A0AFFFF	64	SWO	*Section 80: Debug support (DBG)*
	0x4A090000 - 0x4A09FFFF	64	CTI1	
	0x4A080000 - 0x4A08FFFF	64	CTI0	
	0x4A070000 - 0x4A07FFFF	64	STM registers	
	0x4A060000 - 0x4A06FFFF	64	TSGEN1	
	0x4A050000 - 0x4A05FFFF	64	REPLICATOR	
	0x4A040000 - 0x4A04FFFF	64	TPIU	
	0x4A030000 - 0x4A03FFFF	64	ETR	
	0x4A020000 - 0x4A02FFFF	64	ETF	
	0x4A010000 - 0x4A01FFFF	64	Funnel	
	0x4A000000 - 0x4A00FFFF	64	SoC ROM Table	
AHB5	0x49000000 - 0x49FFFFFF	16384	Reserved	-
	0x48C00000 - 0x48FFFFFF	4096	DDRPHYC	*Section 14.5.194: DDRPHYC register map*
	0x48811000 - 0x48BFFFFF	4028	Reserved	-
	0x48810000 - 0x48810FFF	4	FMC_NAND	External memory access
	0x48804000 - 0x4880FFFF	48	Reserved	-
	0x48800000 - 0x48803FFF	16	CA35SS	*Section 3.6.5: CA35SS SSC register map*
	0x48340000 - 0x487FFFFF	4864	Reserved	-
	0x48300000 - 0x4833FFFF	256	OTG	*Section 74.14.63: OTG register map*
	0x482F0400 - 0x482FFFFF	63	Reserved	-
	0x482F0000 - 0x482F03FF	1	USBH_EHCI	*Section 75.5.12: USBH register map*
	0x482E0400 - 0x482EFFFF	63	Reserved	-
	0x482E0000 - 0x482E03FF	1	USBH_OHCI	*Section 75.5.12: USBH register map*
	0x482D4000 - 0x482DFFFF	48	Reserved	-
	0x482D0000 - 0x482D3FFF	16	ETH2	*Section 77.11.1: Ethernet register maps*
	0x482C4000 - 0x482CFFFF	48	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB5 (cont'd)	0x482C0000 - 0x482C3FFF	16	ETH1	*Section 77.11.1: Ethernet register maps*
	0x48240400 - 0x482BFFFF	511	Reserved	-
	0x48240000 - 0x482403FF	1	SDMMC3	*Section 34.10.26: SDMMC register map*
	0x48230400 - 0x4823FFFF	63	Reserved	-
	0x48230000 - 0x482303FF	1	SDMMC2	*Section 34.10.26: SDMMC register map*
	0x48220400 - 0x4822FFFF	63	Reserved	-
	0x48220000 - 0x482203FF	1	SDMMC1	*Section 34.10.26: SDMMC register map*
	0x48200400 - 0x4821FFFF	127	Reserved	-
	0x48200000 - 0x482003FF	1	FMC registers	*Section 32.12: FMC register map*
APB4	0x480C0400 - 0x481FFFFF	1279	Reserved	-
	0x480C0000 - 0x480C03FF	1	DDRPERFM	*Section 15.5.25: DDRPERFM register map*
	0x48081000 - 0x480BFFFF	252	Reserved	-
	0x48080000 - 0x48080FFF	4	STGENC	*Section 59.4.32: STGEN register map*
	0x48071000 - 0x4807FFFF	60	Reserved	-
	0x48070000 - 0x48070FFF	4	STGENR	*Section 59.4.32: STGEN register map*
	0x48060000 - 0x4806FFFF	64	Reserved	-
	0x48050000 - 0x4805FFFF	64	DDRDBG	*Section 12.6.17: DDRSS register map*
	0x48040000 - 0x4804FFFF	64	DDRCTRL	*Section 13.7.102: DDRCTRL register map*
	0x48031000 - 0x4803FFFF	60	Reserved	-
	0x48030000 - 0x48030FFF	4	DCMIPP	*Section 42.15: DCMIPP register map*
	0x48022000 - 0x4802FFFF	56	Reserved	-
	0x48020000 - 0x48021FFF	8	CSI	*Section 43.10.11: CSI-2 Host and PHY register map*
	0x48010400 - 0x4801FFFF	63	Reserved	-
	0x48010000 - 0x480103FF	1	LTDC	*Section 46.8.58: LTDC register map*
	0x48000000 - 0x4800FFFF	64	Reserved	-
AHB6	0x46200400 - 0x47FFFFFF	30719	Reserved	-
	0x46200000 - 0x462003FF	1	GPIOZ	*Section 22.4.35: GPIO register map*
APB5	0x46080400 - 0x461FFFFF	1532	Reserved	-
	0x46080000 - 0x460803FF	1	I3C3	*Section 66.16.34: I3C register map*
	0x46070400 - 0x4607FFFF	63	Reserved	-
	0x46070000 - 0x460703FF	1	LPTIM5	*Section 58.7.25: LPTIM register map*
	0x46060400 - 0x4606FFFF	63	Reserved	-
	0x46060000 - 0x460603FF	1	LPTIM4	*Section 58.7.25: LPTIM register map*
	0x46050400 - 0x4605FFFF	63	Reserved	-
	0x46050000 - 0x460503FF	1	LPTIM3	*Section 58.7.25: LPTIM register map*
	0x46040400 - 0x4604FFFF	63	Reserved	-
	0x46040000 - 0x460403FF	1	I2C3	*Section 65.9.17: I2C register map*
	0x46030400 - 0x4603FFFF	63	Reserved	-
	0x46030000 - 0x460303FF	1	LPUART1	*Section 68.7.20: LPUART register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB5 (cont'd)	0x46020400 - 0x4602FFFF	63	Reserved	-
	0x46020000 - 0x460203FF	1	SPI6	*Section 69.11.21: SPI/I2S register map*
	0x46010400 - 0x4601FFFF	63	Reserved	-
	0x46010000 - 0x460103FF	1	TAMP	*Section 64.6.30: TAMP register map*
	0x46000400 - 0x4600FFFF	63	Reserved	-
	0x46000000 - 0x460003FF	1	RTC	*Section 63.6.33: RTC register map*
AHB4	0x442D0400 - 0x45FFFFFF	29887	Reserved	-
	0x442D0000 - 0x442D03FF	1 KB	EXTI2	*Section 29.7.86: EXTI2 register map*
	0x442C0400 - 0x442CFFFF	63	Reserved	-
	0x442C0000 - 0x442C03FF	1	GPIOI	*Section 22.4.35: GPIO register map*
	0x442B0400 - 0x442BFFFF	63	Reserved	-
	0x442B0000 - 0x442B03FF	1	GPIOH	*Section 22.4.35: GPIO register map*
	0x442A0400 - 0x442AFFFF	63	Reserved	-
	0x442A0000 - 0x442A03FF	1	GPIOG	*Section 22.4.35: GPIO register map*
	0x44290400 - 0x4429FFFF	63	Reserved	-
	0x44290000 - 0x442903FF	1	GPIOF	*Section 22.4.35: GPIO register map*
	0x44280400 - 0x4428FFFF	63	Reserved	-
	0x44280000 - 0x442803FF	1	GPIOE	*Section 22.4.35: GPIO register map*
	0x44270400 - 0x4427FFFF	63	Reserved	-
	0x44270000 - 0x442703FF	1	GPIOD	*Section 22.4.35: GPIO register map*
	0x44260400 - 0x4426FFFF	63	Reserved	-
	0x44260000 - 0x442603FF	1	GPIOC	*Section 22.4.35: GPIO register map*
	0x44250400 - 0x4425FFFF	63	Reserved	-
	0x44250000 - 0x442503FF	1	GPIOB	*Section 22.4.35: GPIO register map*
	0x44240400 - 0x4424FFFF	63	Reserved	-
	0x44240000 - 0x442403FF	1	GPIOA	*Section 22.4.35: GPIO register map*
	0x44230000 - 0x4423FFFF	64	SYSCFG	*Section 23.2.58: SYSCFG register map*
	0x44220400 - 0x4422FFFF	63	Reserved	-
	0x44220000 - 0x442203FF	1	EXTI1	*Section 29.7.85: EXTI1 register map*
	0x44210400 - 0x4421FFFF	63	Reserved	-
	0x44210000 - 0x442103FF	1	PWR	*Section 19.12.27: PWR register map*
	0x44200000 - 0x4420FFFF	64	RCC	*Section 20.8.206: RCC register map*
APB3	0x440A0400 - 0x441FFFFF	1407	Reserved	-
	0x440A0000 - 0x440A03FF	1	DBGMCU	*Section : DBGMCU register map*
	0x44090400 - 0x4409FFFF	63	Reserved	-
	0x44090000 - 0x440903FF	1	HDP	*Section 78.3.10: HDP register map*
	0x44081000 - 0x4408FFFF	60	Reserved	-
	0x44080000 - 0x44080FFF	4	SERC	*Section 79.6.24: SERC register map*
	0x44071000 - 0x4407FFFF	60	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB3 (cont'd)	0x44070000 - 0x44070FFF	4	DTS	*Section 37.5.5: DTS register map*
	0x44060400 - 0x4406FFFF	63	Reserved	-
	0x44060000 - 0x440603FF	1	VREFBUF	*Section 38.6.3: VREFBUF register map*
	0x44050400 - 0x4405FFFF	63	Reserved	-
	0x44050000 - 0x440503FF	1	WWDG1	*Section 62.6.8: WWDG register map*
	0x44040400 - 0x4404FFFF	63	Reserved	-
	0x44040000 - 0x440403FF	1	IWDG4	*Section 61.7.12: IWDG register map*
	0x44030400 - 0x4403FFFF	63	Reserved	-
	0x44030000 - 0x440303FF	1	IWDG3	*Section 61.7.12: IWDG register map*
	0x44020400 - 0x4402FFFF	63	Reserved	-
	0x44020000 - 0x440203FF	1	IWDG2	*Section 61.7.12: IWDG register map*
	0x44010400 - 0x4401FFFF	63	Reserved	-
	0x44010000 - 0x440103FF	1	IWDG1	*Section 61.7.12: IWDG register map*
	0x44001000 - 0x4400FFFF	60	Reserved	-
	0x44000000 - 0x44000FFF	4	BSEC	*Section 6.5.34: BSEC register map*
AHB3	0x42131000 - 0x43FFFFFF	31548	Reserved	-
	0x42130000 - 0x42130FFF	4	RISAB5	*Section 9.5.20: RISAB register map*
	0x42111000 - 0x4212FFFF	124	Reserved	-
	0x42110000 - 0x42110FFF	4	RISAB3	*Section 9.5.20: RISAB register map*
	0x42101000 - 0x4210FFFF	60	Reserved	-
	0x42100000 - 0x42100FFF	4	RISAB2	*Section 9.5.20: RISAB register map*
	0x420F1000 - 0x420FFFFF	60	Reserved	-
	0x420F0000 - 0x420F0FFF	4	RISAB1	*Section 9.5.20: RISAB register map*
	0x420D1000 - 0x420EFFFF	124	Reserved	-
	0x420D0000 - 0x420D0FFF	4	RISAF4	*Section 10.5.27: RISAF register map*
	0x420B1000 - 0x420CFFFF	124	Reserved	-
	0x420B0000 - 0x420B0FFF	4	RISAF2	*Section 10.5.27: RISAF register map*
	0x420A1000 - 0x420AFFFF	60	Reserved	-
	0x420A0000 - 0x420A0FFF	4	RISAF1	*Section 10.5.27: RISAF register map*
	0x42090400 - 0x4209FFFF	63	Reserved	-
	0x42090000 - 0x420903FF	1	IAC	*Section 11.6.10: IAC register map*
	0x42081000 - 0x4208FFFF	60	Reserved	-
	0x42080000 - 0x42080FFF	4	RIFSC	*Section 8.5.17: RIFSC register map*
	0x42071000 - 0x4207FFFF	60	Reserved	-
	0x42070000 - 0x42070FFF	4	RAMCFG	*Section 16.6.28: RAMCFG register map*
	0x42041000 - 0x4206FFFF	196	Reserved	-
	0x42040000 - 0x42040FFF	4	CRYP2	*Section 49.8.27: CRYP register map*
	0x42039C00 - 0x4203FFFF	25	Reserved	-
	0x42037C00 - 0x42039BFF	8	Reserved	-

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB3 (cont'd)	0x42034000 - 0x42037BFF	15	Reserved	-
	0x42032000 - 0x42033FFF	8	PKA	*Section 52.7.9: PKA register map*
	0x42031000 - 0x42031FFF	4	Reserved	-
	0x42030C00 - 0x42030FFF	1	SAES	*Section 48.8.25: SAES register map*
	0x42030800 - 0x42030BFF	1	RNG1	*Section 47.7.10: RNG register map*
	0x42030400 - 0x420307FF	1	HASH1	*Section 50.6.12: HASH register map*
	0x42030000 - 0x420303FF	1	CRYP1	*Section 49.8.27: CRYP register map*
	0x42020400 - 0x4202FFFF	63	Reserved	-
	0x42020000 - 0x420203FF	1	RNG2	*Section 47.7.10: RNG register map*
	0x42010400 - 0x4201FFFF	63	Reserved	-
	0x42010000 - 0x420103FF	1	HASH2	*Section 50.6.12: HASH register map*
	0x42002000 - 0x4200FFFF	56	Reserved	-
	0x42000000 - 0x42001FFF	8	BKPSRAM	Memory access
AHB2	0x404F0400 - 0x41FFFFFF	27711	Reserved	-
	0x404F0000 - 0x404F03FF	1	ADC2	*Section 36.8: ADC register map*
	0x404E0400 - 0x404EFFFF	63	Reserved	-
	0x404E0000 - 0x404E03FF	1	ADC1	*Section 36.8: ADC register map*
	0x404D1000 - 0x404DFFFF	60	Reserved	-
	0x404D0000 - 0x404D0FFF	4	MDF1	*Section 39.8.25: MDF register map*
	0x404C0400 - 0x404CFFFF	63	Reserved	-
	0x404C0000 - 0x404C03FF	1	CRC	*Section 30.4.6: CRC register map*
	0x404B0400 - 0x404BFFFF	63	Reserved	-
	0x404B0000 - 0x404B03FF	1	PSSI	*Section 44.5.12: PSSI register map*
	0x404A0400 - 0x404AFFFF	63	Reserved	-
	0x404A0000 - 0x404A03FF	1	DCMI	*Section 41.5.16: DCMI register map*
	0x40490400 - 0x4049FFFF	63	Reserved	-
	0x40490000 - 0x404903FF	1	IPCC1	*Section 21.4.19: IPCC register map*
	0x40480400 - 0x4048FFFF	63	Reserved	-
	0x40480000 - 0x404803FF	1	DCACHE	*Section 18.7.15: DCACHE register map*
	0x40470400 - 0x4047FFFF	63	Reserved	-
	0x40470000 - 0x404703FF	1	ICACHE	*Section 17.7.12: ICACHE register map*
	0x40450400 - 0x4046FFFF	127	Reserved	-
	0x40450000 - 0x404503FF	1	OTFDEC1	*Section 51.6.18: OTFDEC register map*
	0x40430400 - 0x4044FFFF	127	Reserved	-
	0x40430000 - 0x404303FF	1	OCTOSPI1	*Section 33.7.33: OCTOSPI register map*
	0x40421000 - 0x4042FFFF	60	Reserved	-
	0x40420000 - 0x40420FFF	4	HPDMA3	*Section 25.8.40: HPDMA register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
AHB2 (cont'd)	0x40411000 - 0x4041FFFF	60	Reserved	-
	0x40410000 - 0x40410FFF	4	HPDMA2	*Section 25.8.40: HPDMA register map*
	0x40401000 - 0x4040FFFF	60	Reserved	-
	0x40400000 - 0x40400FFF	4	HPDMA1	*Section 25.8.40: HPDMA register map*
APB2	0x40370400 - 0x403FFFFF	575	Reserved	-
	0x40370000 - 0x403703FF	1	UART7	*Section 67.8.22: USART register map*
	0x40340400 - 0x4036FFFF	191	Reserved	-
	0x40340000 - 0x403403FF	1	SAI4	*Section 70.6.24: SAI register map*
	0x40330400 - 0x4033FFFF	63	Reserved	-
	0x40330000 - 0x403303FF	1	USART1	*Section 67.8.22: USART register map*
	0x40312800 - 0x4032FFFF	114	Reserved	-
	0x40310000 - 0x403127FF	10	FDCANRAM	Memory access
	0x40300400 - 0x4030FFFF	63	Reserved	-
	0x40300000 - 0x403003FF	1	CCU	*Section 72.7.7: CCU register map*
	0x402E0400 - 0x402FFFFF	127	Reserved	-
	0x402E0000 - 0x402E03FF	1	FDCAN2	*Section 72.5.47: FDCAN register map*
	0x402D0400 - 0x402DFFFF	63	Reserved	
	0x402D0000 - 0x402D03FF	1	FDCAN1	*Section 72.5.47: FDCAN register map*
	0x402B0400 - 0x402CFFFF	127	Reserved	-
	0x402B0000 - 0x402B03FF	1	SAI3	*Section 70.6.24: SAI register map*
	0x402A0400 - 0x402AFFFF	63	Reserved	-
	0x402A0000 - 0x402A03FF	1	SAI2	*Section 70.6.24: SAI register map*
	0x40290400 - 0x4029FFFF	63	Reserved	-
	0x40290000 - 0x402903FF	1	SAI1	*Section 70.6.24: SAI register map*
	0x40280400 - 0x4028FFFF	63	Reserved	
	0x40280000 - 0x402803FF	1	SPI5	*Section 69.11.21: SPI/I2S register map*
	0x40270400 - 0x4027FFFF	63	Reserved	-
	0x40270000 - 0x402703FF	1	TIM17	*Section 57.8.26: TIM16/TIM17 register map*
	0x40260400 - 0x4026FFFF	63	Reserved	-
	0x40260000 - 0x402603FF	1	TIM16	*Section 57.8.26: TIM16/TIM17 register map*
	0x40250400 - 0x4025FFFF	63	Reserved	-
	0x40250000 - 0x402503FF	1	TIM15	*Section 54.5.30: TIMx register map*
	0x40240400 - 0x4024FFFF	63	Reserved	-
	0x40240000 - 0x402403FF	1	SPI4	*Section 69.11.21: SPI/I2S register map*
	0x40230400 - 0x4023FFFF	63	Reserved	-
	0x40230000 - 0x402303FF	1	SPI1	*Section 69.11.21: SPI/I2S register map*
	0x40220400 - 0x4022FFFF	63	Reserved	-
	0x40220000 - 0x402203FF	1	USART6	*Section 67.8.22: USART register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB2 (cont'd)	0x40210400 - 0x4021FFFF	63	Reserved	-
	0x40210000 - 0x402103FF	1	TIM8	*Section 53.6.36: TIMx register map*
	0x40200400 - 0x4020FFFF	63	Reserved	-
	0x40200000 - 0x402003FF	1	TIM1	*Section 53.6.36: TIMx register map*
APB1	0x401D0400 - 0x401FFFFF	191	Reserved	-
	0x401D0000 - 0x401D03FF	1	TIM11	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x401C0400 - 0x401CFFFF	63	Reserved	-
	0x401C0000 - 0x401C03FF	1	TIM10	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x401A0400 - 0x401BFFFF	127	Reserved	-
	0x401A0000 - 0x401A03FF	1	I3C2	*Section 66.16.34: I3C register map*
	0x40190400 - 0x4019FFFF	63	Reserved	-
	0x40190000 - 0x401903FF	1	I3C1	*Section 66.16.34: I3C register map*
	0x40180400 - 0x4018FFFF	63	Reserved	-
	0x40180000 - 0x401803FF	1	I2C2	*Section 65.9.17: I2C register map*
	0x40170400 - 0x4017FFFF	63	Reserved	-
	0x40170000 - 0x401703FF	1	I2C1	*Section 65.9.17: I2C register map*
	0x40110400 - 0x4016FFFF	383	Reserved	-
	0x40110000 - 0x401103FF	1	UART5	*Section 67.8.22: USART register map*
	0x40100400 - 0x4010FFFF	63	Reserved	-
	0x40100000 - 0x401003FF	1	UART4	*Section 67.8.22: USART register map*
	0x400F0400 - 0x400FFFFF	63	Reserved	-
	0x400F0000 - 0x400F03FF	1	USART3	*Section 67.8.22: USART register map*
	0x400E0400 - 0x400EFFFF	63	Reserved	-
	0x400E0000 - 0x400E03FF	1	USART2	*Section 67.8.22: USART register map*
	0x400D0400 - 0x400DFFFF	63	Reserved	-
	0x400D0000 - 0x400D03FF	1	SPDIFRX	*Section 71.5.13: SPDIFRX interface register map*
	0x400C0400 - 0x400CFFFF	63	Reserved	-
	0x400C0000 - 0x400C03FF	1	SPI3	*Section 69.11.21: SPI/I2S register map*
	0x400B0400 - 0x400BFFFF	63	Reserved	-
	0x400B0000 - 0x400B03FF	1	SPI2	*Section 69.11.21: SPI/I2S register map*
	0x400A0400 - 0x400AFFFF	63	Reserved	-
	0x400A0000 - 0x400A03FF	1	LPTIM2	*Section 58.7.25: LPTIM register map*
	0x40090400 - 0x4009FFFF	63	Reserved	-
	0x40090000 - 0x400903FF	1	LPTIM1	*Section 58.7.25: LPTIM register map*
	0x40080400 - 0x4008FFFF	63	Reserved	-
	0x40080000 - 0x400803FF	1	TIM14	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*

Table 9. Register boundary addresses for Cortex-A35 (CPU1) and Cortex-M33 (CPU2)(1) (continued)

Bus	Boundary address	Size (Kbytes)	Peripheral	Peripheral register map
APB1 (cont'd)	0x40070400 - 0x4007FFFF	63	Reserved	-
	0x40070000 - 0x400703FF	1	TIM13	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x40060400 - 0x4006FFFF	63	Reserved	-
	0x40060000 - 0x400603FF	1	TIM12	*Section 56.8.18: TIM10/TIM11/TIM13/TIM14 register map*
	0x40050400 - 0x4005FFFF	63	Reserved	-
	0x40050000 - 0x400503FF	1	TIM7	*Section 55.4.14: TIMx register map*
	0x40040400 - 0x4004FFFF	63	Reserved	-
	0x40040000 - 0x400403FF	1	TIM6	*Section 55.4.14: TIMx register map*
	0x40030400 - 0x4003FFFF	63	Reserved	-
	0x40030000 - 0x400303FF	1	TIM5	*Section 54.5.30: TIMx register map*
	0x40020400 - 0x4002FFFF	63	Reserved	-
	0x40020000 - 0x400203FF	1	TIM4	*Section 54.5.30: TIMx register map*
	0x40010400 - 0x4001FFFF	63	Reserved	-
	0x40010000 - 0x400103FF	1	TIM3	*Section 54.5.30: TIMx register map*
	0x40000400 - 0x4000FFFF	63	Reserved	-
	0x40000000 - 0x400003FF	1	TIM2	*Section 54.5.30: TIMx register map*

1. Reserved areas are highlighted in gray.

3 Arm Cortex-A35 (CA35SS)

3.1 Overview

The Cortex-A35 includes the following features:

- Full implementation of the Armv8-A A64, A32, and T32 instruction sets
- Both AArch32 and AArch64 execution states at all exception levels (EL0 to EL3)
- In order pipeline with direct and indirect branch prediction
- Separate Level 1 (L1) data- and instruction-side memory systems with a memory management unit (MMU)
- Level 2 (L2) memory system that provides Cortex-A35 subsystem memory coherency
- Arm TrustZone
- Advanced SIMD and floating-point architecture support
- Armv8 debug logic
- Performance monitoring unit (PMU)
- Program trace macrocell (PTM) that supports instruction trace only
- Generic timers supporting 64-bit count input from an external system counter

The STM32MP21xx feature a Cortex-A35 subsystem with characteristics detailed in the table below.

Table 10. Cortex-A35 subsystem configuration

Level	Characteristics	Comments
Core level	L1 instruction cache size	32 Kbytes
	L1 data cache size	32 Kbytes
	Cache protection	No
	NEON/FPU	Yes
Subsystem level	Number of processor cores	1
	Master bus interface	AXI4
	Debug memory map	v8
	L2 cache size	128 Kbytes

Some peripherals are associated with the Cortex-A35, and build the Cortex-A35 subsystem:

- Arm GIC (generic interrupt controller): centralizes resources to support and manage interrupts. Its registers manage interrupt sources, interrupt behavior, and interrupt routing to one or more processor cores (see *Section 28: Global interrupt controller (GIC)*).
- STGEN (time generator): provides a time reference for the internal timers of the Cortex-A35 processor core (see *Section 59: System timer generator (STGEN)*).
- system configuration registers (*Section 3.5*) that deal with some static but programmable Cortex-A35 subsystem configuration
- standardized status and control registers (*Section 3.6*) that deal with control and status of subblocks such as Cortex-A35 processor core, PLL, and debug

3.2 Cortex-A35 reset

There are different reset signals that apply to the Cortex-A35 subsystem. The RCC allows some of them to be under software control, as shown in the table below.

Table 11. Cortex-A35 subsystem reset sources

RCC register	Bit	Name	Description
-	-	POR	Device power-on reset (POR)
RCC_GRSTCSETR	0	SYSRST	System reset
RCC_C1RSTCSETR	0	C1RST	Power-on reset for the Cortex-A35 subsystem

The effect of RCC registers on core reset becomes effective only when the Cortex-A35 core enters in idle mode after execution of the wait-for-interrupt (WFI) instruction, ensuring that any transactions on the interconnect are completed. Refer to *Section 20* for more details.

3.3 Cortex-A35 clocking

The clock for the Cortex-A35 subsystem can be selected among:

- a clock from the device clock generator (aka ck_cpu1_ext2f). The maximum frequency on this clock is 400 MHz with a divider by two, enabled thanks to the CA35SS_SSC_CHGCLKREQ SSC register.
- a dedicated PLL that is identical to those embedded in the device clock generator but without the SSCG feature

This selection is done thanks to the CA35SS_SSC_CHGCLKREQ register detailed in *Section 3.6*.

3.4 CortexA35 boot mode

The Cortex-A35 can execute code in either 32 (aarch32) or 64 (aarch64) architecture bit mode.

By default, the Cortex-A35 core boots in the aarch32 bit mode, but this can be changed through a specific procedure conducted by the boot ROM firmware as follows:

1. The core sets the AA64NAA32 bit and RVBARADDRL boot address in dedicated system configuration (CA35SS_SYSCFG) registers detailed in next section.
2. The core requests a warm reset through its internal reset management register in secure mode.
3. The core enters in WFI state.
4. Once this is effective, a warm reset is generated.
5. The core restarts in aarch64 bit mode from the address specified in CA35SS_SYSCFG_VBAR_CR register.

3.5 CA35SS SYSCFG registers

A set of dedicated registers allows the control on static configuration signals for the Cortex-A35 core and a few debug components. They are also controlling the Cortex-M33 core boot parameters as this one is under control of the Cortex-A35 at boot time.

These registers are detailed in the next section. Their address is an offset related to the CA35SS registers base address (refer to *Section 2.4.2*).

These registers have different security access rights that are described in the different following sub-sections along with their meaning.

3.5.1 System debug configuration registers

The following registers allow the programming of system debug registers. They are not reserved to a particular master, and as such, are not subject to access restrictions. They are reset only by a power-on reset.

CA35SS ETR low-power interface control register (CA35SS_SYSCFG_ETR_LPI_CR)

Address offset: 0x2000

Reset value: 0x0000 0001

This register controls the low-power interface of the embedded trace router.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR_CSYSREQ
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ETR_CSYSREQ**:

0: Request ETR AXI bus to be stalled (csysreq is active low).

1: Request ETR AXI activity to resume.

CA35SS ETR low-power interface status register (CA35SS_SYSCFG_ETR_LPI_SR)

Address offset: 0x2004

Reset value: 0x0000 0000

This register returns the status of the low-power interface of the embedded trace router.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR_CSYSACK
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ETR_CSYSACK**:

0: ETR AXI transactions stalled

1: ETR AXI transactions not stalled

CA35SS STM non-secure guaranteed access register (CA35SS_SYSCFG_STM_NSGUAREN_CR)

Address offset: 0x2008

Reset value: 0x0000 0001

This register controls the STM behavior for non-secure accesses on its stimulus port.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STM_NSGUAREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **STM_NSGUAREN**:

0: Non-secure access is not guaranteed to be traced. The AXI write data channel does not stall when the STM is unable to trace the transaction immediately. The STM discards write data if it cannot be traced.

1: Non-secure access is guaranteed to be traced, that is the AXI write data channel stalls until the STM can emit the trace. This is obviously intrusive with respect to the real time.

CA35SS TPIU export clock divider register (CA35SS_SYSCFG_TRACE_CLK_DIV_CR)

Address offset: 0x200C

Reset value: 0x0000 0000

This register allows the trace port interface unit export clock trace (ck_ker_tpiu) to be reduced further to what is delivered at the flexible clock generator channel number 44.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_CLK_DIV_VALUE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRACE_CLK_DIV_VALUE[5:0]**:
0: Divider bypassed
0x1 to 0x3F: Divider ratio from 2 to 64

CA35SS Cortex-A35 core debug power mode register (CA35SS_SYSCFG_DBGPWR_CR)

Address offset: 0x2010

Reset value: 0x0000 0003

This register controls the power on Cortex-A35 debug resources by preventing any external debug to access a core that is powered down.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGPWRUP0
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBGPWRUP0**:
0: Core 0 debug powered down
1: Core 0 debug powered up

CA35SS Cortex-A35 L1 data cache register (CA35SS_SYSCFG_DBGL1RSTDISABLE_CR)

Address offset: 0x2014

Reset value: 0x0000 0000

This register controls the behavior of the L1 data cache at processor reset. This can be used when debugging an external reset as this makes the contents of the L1 data cache from before the reset observable after the reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGL1RSTDIS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBGL1RSTDIS**:

0: Enable automatic invalidation of L1 data cache on reset.

1: Disable automatic invalidation of L1 data cache on reset.

CA35SS Cortex-A35 debug power register (CA35SS_SYSCFG_DBGPWR_SR)

Address offset: 0x2018

Reset value: 0x0000 0000

This register returns information on debug requests with respect to power management.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGPWRUPREQ0	Res.	DBGNOPWRDWN0
													r		r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DBGPWRUPREQ0**:

0: Debug does not request that the core 0 is powered-up.

1: Debug requests that the core 0 is powered-up.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DBGNOPWRDWN0**:

0: Debug does not request that the core 0 stays powered-up.

1: Debug requests that the core 0 stays powered-up.

CA35SS Cortex-A35 debug mode status register (CA35SS_SYSCFG_EDBGACK_SR)

Address offset: 0x201C

Reset value: 0x0000 0000

This register returns information on whether a core has entered in debug mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGACK_C0
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBGACK_C0**:
0: Core 0 is not in debug mode.
1: Core 0 is halted in debug mode.

3.5.2 Cortex-A35 subsystem configuration registers

These registers are used to modify the GIC setup, and to provide some information on Cortex-A35 states.

They are only accessible to a master with a CID = 1 (such as the Cortex-A35 or the debugger when Cortex-A35 debug is enabled). No secure or privilege rights apply to them.

These registers are reset by the power-on reset, the application reset or C1RST in the RCC.

GIC configuration register (CA35SS_SYSCFG_GIC_CFGR)

Address offset: 0x2020

Reset value: 0x0000 0000

This register allows the GIC to be used in a 64-Kbyte translation, so that the GICC register frame is spread across two MMU pages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GIC_400_CFG_DISABLE	GIC_4KNOT64K
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **GIC_400_CFG_DISABLE**:
0: GIC400 secure registers can be written
1: GIC400 secure registers locked

Bit 0 **GIC_4KNOT64K**:
0: Use the GIC in a 64-Kbyte translation granule system.
1: Use the GIC in a 4-Kbyte translation granule system.

Cortex-A35 low-power status register (CA35SS_SYSCFG_LP_SR)

Address offset: 0x2024

Reset value: 0x0000 0000

This register returns information on low-power states for core and L2 cache.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STANDBYWFIL2	Res.	STANDBYWFI_0	Res.	STANDBYWFE_0
											r		r		r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **STANDBYWFIL2:**
0: L2 memory system not in WFI low-power state
1: L2 memory system in WFI low-power state

Bit 3 Reserved, must be kept at reset value.

Bit 2 **STANDBYWFI_0**:
0: Core 0 not in WFI low-power state
1: Core 0 in WFI low-power state

Bit 1 Reserved, must be kept at reset value.

Bit 0 **STANDBYWFE_0**:
0: Core 0 not in WFE low-power state
1: Core 0 in WFE low-power state

Cortex-A35 reset status register (CA35SS_SYSCFG_RSTACK_SR)

Address offset: 0x2028

Reset value: 0x0000 0000

This register returns information on reset states.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NRESET_FCT_ACK	NRESET_DBG_ACK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NRESET_FCT_ACK**:
- 0: Functional reset active
- 1: Functional reset inactive

Bit 0 **NRESET_DBG_ACK**:
- 0: Debug reset active
- 1: Debug reset inactive

Cortex-A35 configuration register (CA35SS_SYSCFG_CP15_CFGR)

Address offset: 0x202C

Reset value: 0x0000 0000

This register configures the access to Cortex-A35 registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CP15_DISABLE_CORE_0
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CP15_DISABLE_CORE_0**:
- 0: access allowed
- 1: disable access to Cortex-A35 CP15 system control coprocessor core 0

3.5.3 Cortex-A35 boot configuration registers

These registers are used to modify the architecture state of the Cortex-A35 core. They are only accessible to the trusted master, thus protected by the TD_CID, with secure and

privilege rights. They are reset by the power-on reset, the application reset or C1RST in the RCC.

CA35SS architecture mode register (CA35SS_SYSCFG_AARCH_MODE_CR)

Address offset: 0x2080

Reset value: 0x0000 0000

This register forces the architecture mode at reset exit for the core. This is a write once register so that the architecture mode is locked.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA64NAA32
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AA64NAA32**:
0: Cortex-A35 core boots in 32-bit mode.
1: Cortex-A35 core boots in 64-bit mode.

CA35SS reset vector register (CA35SS_SYSCFG_VBAR_CR)

Address offset: 0x2084

Reset value: 0x0000 0000

This register forces the reset vector base address in 64-bit mode. Its value is sampled when the core is released from reset. It is not applicable for 32-bit architecture mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RVBARADDRL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVBARADDRL[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **RVBARADDRL[31:2]:** Set the value of reset vector at the boundaries of the Cortex-A35 core.

Bits 1:0 Reserved, must be kept at reset value.

3.5.4 Cortex-M33 boot configuration registers

These registers are used to setup the boot configuration of the Cortex-M33. They are only accessible to the trusted master thus protected by the TD_CID.

CA35SS_SYSCFG_M33_ACCESS_CR register sets secure and/or privilege rights to access the Cortex-M33 boot options. It must be accessed with secure and privilege rights.

These registers are reset by the power-on reset or an application reset.

Cortex-M33 boot register access rights register (CA35SS_SYSCFG_M33_ACCESS_CR)

Address offset: 0x2088

Reset value: 0x0000 0003

This register sets up the access conditions on the boot options.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	M33CFG_PRIV	M33CFG_SEC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **M33CFG_PRIV**:
0: Cortex-M33 boot option registers are accessed in non-privilege and privilege modes.
1: Cortex-M33 boot option registers are accessed in privilege mode only.

Bit 0 **M33CFG_SEC**:
0: Cortex-M33 boot option registers are accessed in non-secure and secure modes.
1: Cortex-M33 boot option registers are accessed in secure mode only.

Cortex-M33 secure extension enable register (CA35SS_SYSCFG_M33_TZEN_CR)

Address offset: 0x20A0

Reset value: 0x0000 0001

This register enables support for Armv8-M security extension in Cortex-M33.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG_SECEXT
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CFG_SECEXT**:
0: Armv8-M TrustZone support disabled
1: Armv8-M TrustZone support enabled

Cortex-M33 secure vector address register (CA35SS_SYSCFG_M33_INITSVTOR_CR)

Address offset: 0x20A4

Reset value: 0x0E04 0000

This register sets the Cortex-M33 secure vector table base address. The reset value locates this table in RETRAM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INITSVTOR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITSVTOR[15:7]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **INITSVTOR[31:7]:** Secure vector table offset

Bits 6:0 Reserved, must be kept at reset value.

Cortex-M33 non-secure vector address register (CA35SS_SYSCFG_M33_INITNSVTOR_CR)

Address offset: 0x20A8

Reset value: 0x0A04 0000

This register sets the Cortex-M33 non-secure vector table base address. The reset value locates the table in RETRAM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INITNSVTOR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITNSVTOR[15:7]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **INITNSVTOR[31:7]:** Non-secure vector table offset

Bits 6:0 Reserved, must be kept at reset value.

3.5.5 CA35SS SYSCFG register map

Table 12. CA35SS SYSCFG register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2000	**CA35SS_SYSCFG_ETR_LPI_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR_CSYSREQ
	Reset value																																1
0x2004	**CA35SS_SYSCFG_ETR_LPI_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR_CSYSACK
	Reset value																																0
0x2008	**CA35SS_SYSCFG_STM_NSGUAREN_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STM_NSGUAREN
	Reset value																																1
0x200C	**CA35SS_SYSCFG_TRACE_CLK_DIV_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_CLK_DIV_VALUE[5:0]					
	Reset value																											0	0	0	0	0	0
0x2010	**CA35SS_SYSCFG_DBGPWR_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGPWRUP0
	Reset value																																1
0x2014	**CA35SS_SYSCFG_DBGL1RSTDISABLE_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGL1RSTDIS
	Reset value																																0
0x2018	**CA35SS_SYSCFG_DBGPWR_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGPWRUPREQ0	Res.	DBGNOPWRDWN0
	Reset value																														0		0
0x201C	**CA35SS_SYSCFG_EDBGACK_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGACK_C0
	Reset value																																0

Table 12. CA35SS SYSCFG register map and reset value (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2020	**CA35SS_SYSCFG_GIC_CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GIC_400_CFG_DISABLE.	GIC_4KNOT64K.
	Reset value																															0	0
0x2024	**CA35SS_SYSCFG_LP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STANDBYWFIL2	Res.	STANDBYWFI_0	Res.	STANDBYWFE_0
	Reset value																												0		0		0
0x2028	**CA35SS_SYSCFG_RSTACK_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NRESET_FCT_ACK	NRESET_DBG_ACK
	Reset value																															0	0
0x202C	**CA35SS_SYSCFG_CP15_CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CP15_DISABLE_CORE_0
	Reset value																																0
0x2030-0x207C	Reserved	Reserved																															
0x2080	**CA35SS_SYSCFG_AARCH_MODE_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA64NAA32
	Reset value																																0
0x2084	**CA35SS_SYSCFG_VBAR_CR**	RVBARADDRL[31:2]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x2088	**CA35SS_SYSCFG_M33_ACCESS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	M33CFG_PRIV	M33CFG_SEC
	Reset value																															1	1
0x208C-0x209C	Reserved	Reserved																															

Table 12. CA35SS SYSCFG register map and reset value (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20A0	**CA35SS_SYSCFG_M33_TZEN_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG_SECEXT
	Reset value																																1
0x20A4	**CA35SS_SYSCFG_M33_INITSVTOR_CR**	INITSVTOR[31:7]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0							
0x20A8	**CA35SS_SYSCFG_M33_INITNSVTOR_CR**	INITNSVTOR[31:7]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0							

3.6 Standardized status and control registers

Another set of registers is implemented to control and monitor some features tightly coupled with the Cortex-A35 subsystem.

3.6.1 Standardized status and control registers (SSC) access modes

The SSC addresses offset are related to the CA35SS register base address (Refer to *Section 2.4.2*). There are four addresses for each SSC register.

Different actions are performed when writing the registers with each of the four addresses. Each register address is given in the ‘Address offset’ line by a formula such as: <reg_address> + 0x4 * x (x = 0 to 3), and additional information is given through the ‘Type’ line, for example RW, WS1, WC1, WT1:

- Writing data at address <reg_address> + 0x0 is a standard write of all data bits to all bits of the register.
- Writing data at address <reg_address> + 0x4 is a 'write set bit': all data bits at 1 set corresponding register bits to 1. Other register bits are unchanged.
- Writing data at address <reg_address> + 0x8 is a 'write clear bit': all data bits at 1 reset corresponding register bits to 0. Other register bits are unchanged.
- Writing data at address <reg_address> + 0xC is a 'write toggle bit': all data bits at 1 toggle corresponding register bits to their complementary value. Other register bits are unchanged.

Reading at any of the four addresses returns the register value to the extent that the secure access right defined for the register is matched. Otherwise, it returns zero when read.

3.6.2 SSC access types

The SSC registers are accessible only from a master with a CID = 1 (for example the Cortex-A35 or the debugger when Cortex-A35 debug is enabled). They can only be accessed in non-burst 4-byte mode. Any other access type is not authorized.

3.6.3 Secure/non-secure access to SSC

By default all SSC registers must be accessed in secure mode, but this can be changed as explained here below.

The global security control of SSC registers is achieved by setting one SSC register CA35SS_SSC_NS_ENABLE_0. This register as well as CA35SS_SSC_NS_ENABLE_1 can only be accessed in secure mode, in all cases.

For convenience, these two registers are called security registers.

Inside the SSC register set, each of all the other registers has a flag to control its accessibility from non-secure mode through CA35SS_SSC_NS_ENABLE_0. This security register is located at position 60 in the SSC register map. The registers associated with the bits are shown in table below:

Table 13. Mapping SSC registers to secure flag register

Register number	Address offset	Flag register	Flag register bit position
0	0x0000	CA35SS_SSC_NS_ENABLE_0	0
1	0x0010		1
...	...		...
31	0x01F0		31
60	0x03C0	Always treated as secure	
61	0x03D0		
62	0x03E0	Reserved	
63	0x03F0	Always treated as non-secure	

3.6.4 SSC registers details

CA35SS SSC CHGCLKREQ register (CA35SS_SSC_CHGCLKREQx)

Address offset: 0x000 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0003 0003

This register allows the selection of the Cortex-A35 clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARM_DIVSELACK	ARM_DIVSEL
														r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARM_CHGCLKACK	ARM_CHGCLKREQ
														r	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **ARM_DIVSELACK**:
Defines if ck_cpu_ext2f clock is divided by 2 or not.
0: ck_cpu1_ext2f divided by two
1: ck_cpu1_ext2f not divided by two

Bit 16 **ARM_DIVSEL**:
Selects divider by 2 for ck_cpu1_ext2f clock.
0: Divide external clock ck_cpu1_ext2f by two.
1: Do not divide external clock ck_cpu1_ext2f.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **ARM_CHGCLKACK**:
Indicates clock source.
0: Cortex-A35 clock is the PLL.
1: Cortex-A35 clock is the external device clock.

Bit 0 **ARM_CHGCLKREQ**:
Selects clock source.
0: Select internal PLL for Cortex-A35 clock.
1: Select external clock ck_cpu1_ext2f for Cortex-A35 clock.

CA35SS SSC BRM register (CA35SS_SSC_BRMx)

Address offset: 0x010 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0300 2000

When the core is in WFI and the L2 cache is idle, the binary rate multiplier (BRM) is automatically dividing the main clock of the Cortex-A35 and its derived clocks to save dynamic power. The clock division ends when the CPU or the L2 cache leaves its idle state.

This register configures the BRM of the Cortex-A35 clock (ck_cpu1_ext2f or ck_cpu1_ext2f/2 or Cortex-A35 PLL).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BRM_IDLEDIV[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRM_IDLEDIVEN
						rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BRM_RATE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	BRM_RATEACK	BRM_RATEVAL
		rw	rw	rw	rw	rw	rw							r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **BRM_IDLEDIV[1:0]**: BRM rate when idle mode is entered
Must not be modified when BRM_IDLEDIVEN = 1.
00: CPU clock = BRM clock input / 4
01: CPU clock = BRM clock input / 8
10: CPU clock = BRM clock input / 16
11: CPU clock = BRM clock input / 32

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BRM_IDLEDIVEN**:
Defines BRM_RATE behaviour in idle mode.
0: BRM_RATE remains unchanged when the idle mode is entered.
1: BRM_RATE switches to the BRM_IDLEDIV when the idle mode is entered [CPU0(WFI or WFE) and CPU1(WFI or WFE) and L2 idle mode].

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **BRM_RATE[5:0]**: BRM rate in normal or idle mode when BRM_IDLEDIVEN = 0
1XXXXX: CPU clock = BRM input clock (default reset value)
0YYYYY: CPU clock = BRM input clock * (YYYYY / 32). YYYYY = 00000 is forbidden.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BRM_RATEACK**: BRM_RATE status
0: BRM ready for a new setting
1: BRM new setting acknowledged

Bit 0 **BRM_RATEVAL**: BRM_RATE operation
0: BRM module is ready for a new setting.
1: Forces the BRM module to sample a new setting.

CA35SS SSC PLL FREQ1 register (CA35SS_SSC_PLL_FREQ1_x)

Address offset: 0x080 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0000

This register configures the PLL for the Cortex-A35 clock (see *Section 20.4.5: PLL description* in the RCC for details).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	FBDIV[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **FREFDIV[5:0]**: Reference divide value

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **FBDIV[11:0]**: Feedback divide value

CA35SS SSC PLL FREQ2 register (CA35SS_SSC_PLL_FREQ2_x)

Address offset: 0x090 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0000

This register configures the PLL for the Cortex-A35 clock (see *Section 20.4.5: PLL description* in the RCC for details).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2[2:0]			POSTDIV1[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **POSTDIV2[2:0]**: PLL post divide 2 setting (1 to 7)
POSTDIV2 = 0 is unused.

Bits 2:0 **POSTDIV1[2:0]**: PLL post divide 1 setting (1 to 7)
POSTDIV1 = 0 is unused.

CA35SS SSC PLL enable register (CA35SS_SSC_PLL_ENx)

Address offset: 0x0A0 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0004

This register configures the PLL for the Cortex-A35 clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_PLL_OBS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NRESET_SWPLL	LOCKP	PLL_EN
													rw	r	rw

Bit 31 **EN_PLL_OBS**: PLL observation output status
0: PLL is not observable.
1: Enable PLL output observation in functional mode (PLL clock is divided by 42).

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **NRESET_SWPLL**: Clock switcher reset state
0: Clock switcher logic clocked by the PLL under reset
1: Clock switcher logic not in reset

Bit 1 **LOCKP**: PLLL lock state
0: PLL not locked
1: PLL locked

Bit 0 **PLL_EN**: PLL enable for Cortex-A35
0: PLL disabled
1: PLL enabled

CA35SS SSC LPI TSGEN NTS CR register (CA35SS_SSC_LPI_TSGEN_NTS_CRx)

Address offset: 0x0D0 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0300

This register configures the low-power Interface for the trace time stamp asynchronous bridge used between the device timestamp generator and the Cortex-A35 PTM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TS_CSYSACK	TS_CSYSREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r	rw								

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TS_CSYSACK**: Timestamp asynchronous bridge power status
0: Timestamp asynchronous bridge powered down
1: Timestamp asynchronous bridge powered up

Bit 8 **TS_CSYSREQ**: Timestamp asynchronous bridge power request
0: Request timestamp asynchronous bridge to be powered down (csysreq is active low).
1: Request timestamp asynchronous bridge to be powered up.

Bits 7:0 Reserved, must be kept at reset value.

CA35SS SSC LPI STGEN NTS CR register (CA35SS_SSC_LPI_STGEN_NTS_CRx)

Address offset: 0x140 + 0x4 *x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0300 0000

This register configures the low-power interface for the time generator asynchronous bridge used between the system time generator (STGEN) and the Cortex-A35 internal timers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	STGEN_CSYSACK	STGEN_CSYSREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **STGEN_CSYSACK**: STGEN asynchronous bridge power status
0: STGEN asynchronous bridge is powered down.
1: STGEN asynchronous bridge is powered up.

Bit 24 **STGEN_CSYSREQ**: STGEN asynchronous bridge power request
0: Request STGEN asynchronous bridge to be powered down (csysreq is active low).
1: Request STGEN asynchronous bridge to be powered up.

Bits 23:0 Reserved, must be kept at reset value.

CA35SS SSC OPP REQ register (CA35SS_SSC_OPP_REQx)

Address offset: 0x1A0+ 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0000

This register allows to control the operating point change request and acknowledge for Cortex-A35. This register must be controlled with a specific sequence with other registers or peripherals such as V_{DDCPU} voltage and Cortex-A35 clock controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPP_ACK	OPP_REQ
														r	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OPP_ACK**: Operating point change request status
0: OPP_REQ not active or not acknowledged
1: OPP_REQ acknowledged

Bit 0 **OPP_REQ**: Request
0: Ddo not request an operating point change request to the Cortex-A35 subsystem.
1: Request an operating point change request to the Cortex-A35 subsystem.

CA35SS SSC memory control register (CA35SS_SSC_MEM_CTRLx)

Address offset: 0x1C0+ 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0052

This register control the speed of memories inside Cortex-A35. This register must be controlled with a specific sequence with other registers or peripherals such as V_{DDCPU} voltage and Cortex-A35 clock controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RME	RM[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RME**: Cortex-A35 Memory Read-Write margin enable.
0: default margin setting. Default for nominal Cortex-A35 frequencies.
1: margin setting from RM[2:0]. Recommended for overdrive Cortex-A35 frequencies.

Bits 2:0 **RM[2:0]**: Cortex-A35 memory Read-Write margin. Ignored if RME=0.
3: recommended value for Cortex-A35 overdrive frequencies.

CA35SS SSC NS enable 0 register (CA35SS_SSC_NS_E0Rx)

Address offset: 0x3C0 + 0x4 * x (x = 0 to 3)

Type: RW, WS1, WC1, WT1

Reset value: 0x0000 0000

This register controls the secure/non-secure setting of each mapped SSC register from 0x000 to 0x1F0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALLOW_NONSEC31	ALLOW_NONSEC30	ALLOW_NONSEC29	ALLOW_NONSEC28	ALLOW_NONSEC27	ALLOW_NONSEC26	ALLOW_NONSEC25	ALLOW_NONSEC24	ALLOW_NONSEC23	ALLOW_NONSEC22	ALLOW_NONSEC21	ALLOW_NONSEC20	ALLOW_NONSEC19	ALLOW_NONSEC18	ALLOW_NONSEC17	ALLOW_NONSEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALLOW_NONSEC15	ALLOW_NONSEC14	ALLOW_NONSEC13	ALLOW_NONSEC12	ALLOW_NONSEC11	ALLOW_NONSEC10	ALLOW_NONSEC9	ALLOW_NONSEC8	ALLOW_NONSEC7	ALLOW_NONSEC6	ALLOW_NONSEC5	ALLOW_NONSEC4	ALLOW_NONSEC3	ALLOW_NONSEC2	ALLOW_NONSEC1	ALLOW_NONSEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALLOW_NONSECn**: SSC register security settings (n = 31 to 0)
Bit[n] configures the security settings of the SSC register at address 16 * n up to 0x1F0.
0: SSC register access is secure.
1: SSC register access is non-secure.

CA35SS SSC NS enable 1 register (CA35SS_SSC_NS_ENABLE_1)

Address offset: 0x3D0

Reset value: 0x8000 0000

This register returns the secure/non-secure setting of each mapped SSC register from 0x3C0 to 0x3F0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALLOW_NONSEC_RO3	ALLOW_NONSEC_RO2	ALLOW_NONSEC_RO1	ALLOW_NONSEC_RO0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 **ALLOW_NONSEC_ROn**: SSC register security settings (n = 3 to 0)
Bit[n] indicates the security settings of the SSC register from 0x3C0 (bit 28) up to 0x3F0 (bit 31).
0: SSC register access is secure.
1: SSC register access is non-secure.

Bits 27:0 Reserved, must be kept at reset value.

3.6.5 CA35SS SSC register map

Table 14. CA35SS SSC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4*x (x=0 to 3) Last address: 0x00C	**CA35SS_SSC_CHGCLKREQx**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARM_DIVSELACK	ARM_DIVSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARM_CHGCLKACK	ARM_CHGCLKREQ
	Reset value															1	1															1	1
0x010 + 0x4*x (x=0 to 3) Last address: 0x01C	**CA35SS_SSC_BRMx**	Res.	Res.	Res.	Res.	Res.	Res.	BRM_IDLEDIV[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRM_IDLEDIVEN	Res.	Res.	BRM_RATE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	BRM_RATEACK	BRM_RATEVAL
	Reset value							1	1								0			0	1	0	0	0	0							0	0

Table 14. CA35SS SSC register map and reset value (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080 + 0x4*x (x=0 to 3) Last address: 0x08C	**CA35SS_SSC_PLL_FREQ1_x**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]						Res.	Res.	Res.	Res.	FBDIV[11:0]											
	Reset value											0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x090 + 0x4*x (x=0 to 3) Last address: 0x09C	**CA35SS_SSC_PLL_FREQ2_x**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2 [2:0]			POSTDIV1 [2:0]		
	Reset value																											0	0	0	0	0	0
0x0A0 + 0x4*x (x=0 to 3) Last address: 0x0AC	**CA35SS_SSC_PLL_ENx**	EN_PLL_OBS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NRESET_SWPLL	LOCKP	PLL_EN
	Reset value	0																													1	0	0
0x0B0-0x0CC	Reserved	Reserved																															
0x0D0 + 0x4*x (x=0 to 3) Last address: 0x0DC	**CA35SS_SSC_LPI_TSGEN_NTS_CRx**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_CSYSACK	TS_CSYSREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							1	1								
0x100-0x13C	Reserved	Reserved																															
0x140 + 0x4*x (x=0 to 3) Last address: 0x14C	**CA35SS_SSC_LPI_STGEN_NTS_CRx**	Res.	Res.	Res.	Res.	Res.	Res.	STGEN_CSYSACK	STGEN_CSYSREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							1	1																								
0x150-0x19C	Reserved	Reserved																															
0x01A0 + 0x4*x (x=0 to 3) Last address: 0x1AC	**CA35SS_SSC_OPP_REQx**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPP_ACK	OPP_REQ
	Reset value																															0	0
0x1B0-0x1BC	Reserved	Reserved																															
0x1C0 + 0x4*x (x=0 to 3) Last address: 0x1CC	**CA35SS_SSC_MEM_CTRLx**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RME	RM[2:0]		
	Reset value																													0	0	1	0
0x1D0-0x3BC	Reserved	Reserved																															

Table 14. CA35SS SSC register map and reset value (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C0 + 0x4*x (x=0 to 3) Last address: 0x3CC	**CA35SS_SSC_NS_ENABLE_0_RW**	ALLOW_NONSEC31	ALLOW_NONSEC30	ALLOW_NONSEC29	ALLOW_NONSEC28	ALLOW_NONSEC27	ALLOW_NONSEC26	ALLOW_NONSEC25	ALLOW_NONSEC24	ALLOW_NONSEC23	ALLOW_NONSEC22	ALLOW_NONSEC21	ALLOW_NONSEC20	ALLOW_NONSEC19	ALLOW_NONSEC18	ALLOW_NONSEC17	ALLOW_NONSEC16	ALLOW_NONSEC15	ALLOW_NONSEC14	ALLOW_NONSEC13	ALLOW_NONSEC12	ALLOW_NONSEC11	ALLOW_NONSEC10	ALLOW_NONSEC9	ALLOW_NONSEC8	ALLOW_NONSEC7	ALLOW_NONSEC6	ALLOW_NONSEC5	ALLOW_NONSEC4	ALLOW_NONSEC3	ALLOW_NONSEC2	ALLOW_NONSEC1	ALLOW_NONSEC0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3D0	**CA35SS_SSC_NS_ENABLE_1**	ALLOW_NONSEC_RO3	ALLOW_NONSEC_RO2	ALLOW_NONSEC_RO1	ALLOW_NONSEC_RO0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0																												

Refer to *Section 2.4* for the register boundary addresses.

4 Arm Cortex-M33 (CM33SS)

4.1 Overview

The Cortex-M33 processor is based on the Armv8-M architecture.

The processor supports the following interfaces:

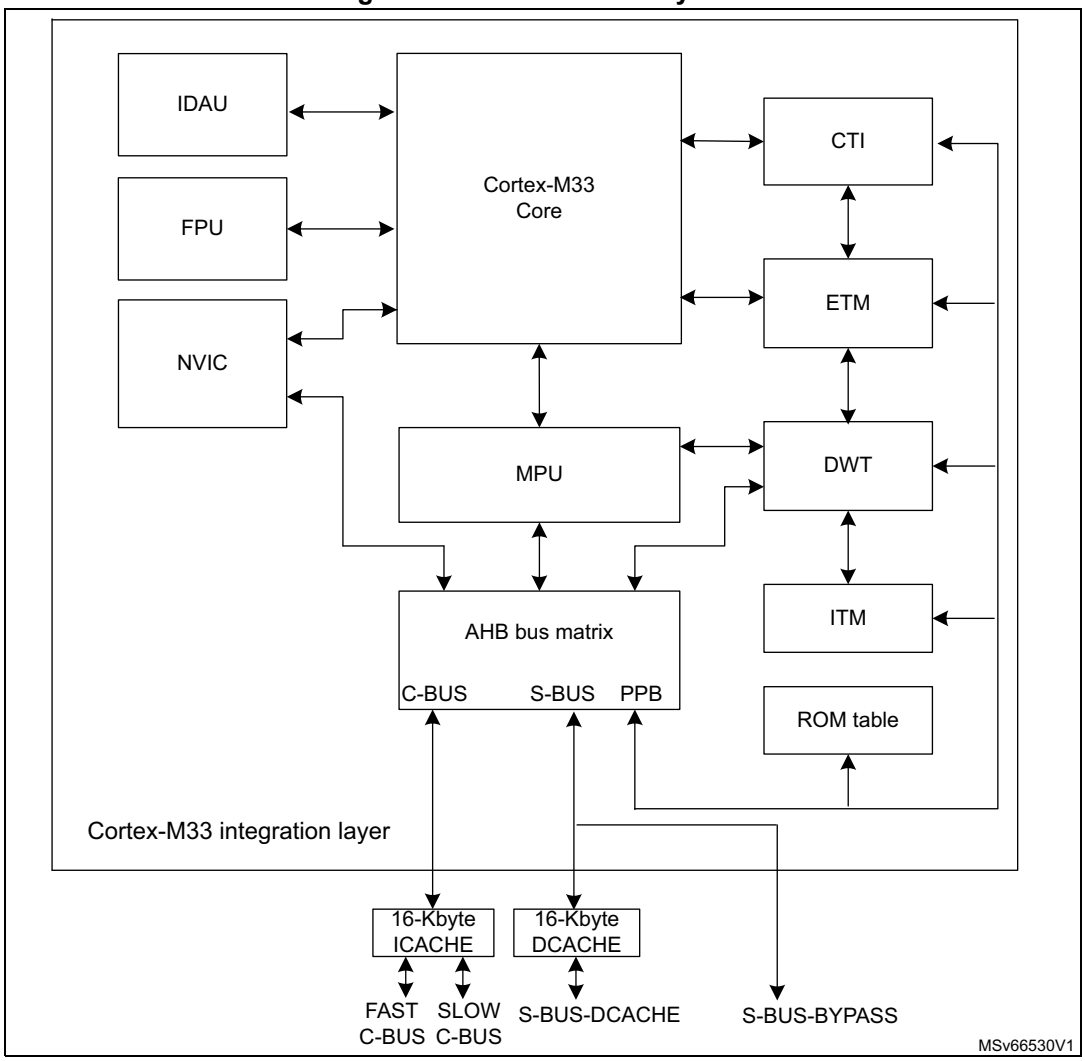
- Code AHB (C-AHB) interface
- System AHB (S-AHB) interface
- External PPB (EPPB) APB interface
- Debug AHB (D-AHB) interface

The STM32MP21xx implementation of the Cortex-M33 supports:

- the Armv8-M security extension
- the floating-point extension
- the Armv8-M DSP extension
- a nested vectored interrupt controller (NVIC) closely integrated with the processor with 320 interrupts and 16 levels of priority
- a memory system with memory protection unit (MPU) with up to 16 non-secure regions and 16 secure regions
- a security attribution unit (SAU) with up to eight regions
- an implementation defined attribution unit (IDAU)
- debug components including breakpoints (BPU), data watchpoints (DWT), instrumentation and processor trace (ITM/ETM), and cross trigger interface (CTI)

16-Kbyte instruction and 16-Kbyte data caches (ICACHE/DCACHE) are added respectively on C-AHB and S-AHB interfaces as shown in *Figure 6*.

Figure 6. Cortex-M33 sub-system



For information on the Arm Cortex-M33 core, refer to the Cortex-M33 r1p0 Technical Reference Manual available at http://infocenter.arm.com.

See *Section 17: Instruction cache (ICACHE)* and *Section 18: Data cache (DCACHE)* for more details.

The debug logic for the Cortex-M33 and its integration within the device are detailed in *Section 80: Debug support (DBG)*.

4.2 Cortex-M33 reset

There are different reset signals that apply to the Cortex-M33 subsystem.

The RCC allows some of them to be under control of the software. The following table indicates these different reset sources.

Table 15. Cortex-M33 reset sources

RCC register	Bit	Name	Description
-	-	POR	Device power-on reset (POR)
RCC_DBGCFGR	12	DBGRST	Reset the Cortex-M33 debug logic
RCC_C2RSTCSETR	0	C2RST	Reset all Cortex-M33 resources but the debug logic
-	-	SYSRESETREQ	Request for functional reset generated by AIRCR core register

4.3 Cortex-M33 clocking

A single clock for the Cortex-M33 system is provided from the device clock generator (ck_cpu2). This clock can be adjusted and gated automatically with respect to the low-power mode of the core.

The RCC feeds the Cortex-M33 system timer (SysTick) external clock with the core clock divided by eight. The SysTick can work either with this clock or directly with the Cortex-M33 clock configurable in the SysTick control and status register. The calibration value is unknown, and the software must compute it from the RCC registers setting up the Cortex-M33 clock.

4.4 Cortex-M33 boot mode

The Cortex-M33 can boot with the Armv8-M security extension ON or OFF depending on its input configuration signal CFGSECEXT is high or low.

This signal is driven from the M33_TZEN_CR register detailed in *Section 3.5.4: Cortex-M33 boot configuration registers*. Restriction applies when accessing this register to comply with security rules.

With the Armv8-M security extension ON, the Cortex-M33 uses two vector tables: one for the secure exceptions, and one for the non-secure exceptions. These vector tables are controlled from the Cortex-M33_INITSVTOR_CR and Cortex-M33_INITNSVTOR_CR registers detailed in *Section 3.5.4: Cortex-M33 boot configuration registers*.

4.5 Cortex-M33 configuration register

A system configuration register SYSCFG_M33SSCR detailed in *Section 3.5: CA35SS SYSCFG registers* is used to prevent changes to occur on SAU/MPU/Vector, and to mask FPU interrupts.

4.6 Memory protection unit (MPU)

The MPU defines the memory attributes for different memory regions. It provides up to 16 non-secure regions and 16 secure regions with the Armv8-M security extension on.

In particular, the cacheable memory regions are defined and programmed by the user in the MPU that is responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

4.7 Floating point unit (FPU)

The Cortex-M33 supports a single-precision FPU that fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. The FPU also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

The FPU can generate six different interrupts that are combined into a single interrupt connected to the NVIC. The SYSCFG_M33SSCR register allows the different FPU interrupts to be masked individually.

4.8 Attribution units

4.8.1 Security attribution unit (SAU)

The SAU is a programmable unit that determines the security of an address. The SAU is enabled when the Armv8-M security extension is on. It provides up to eight different regions, and a default background region.

At reset, before any SAU region is programmed and apart from specific region in the PPB space, all memories are set to secure state.

Secure (S)

Secure addresses are used for memory and peripherals that are only accessible by secure software or secure masters.

Non-secure callable (NSC)

NSC is a special type of secure location. This type of memory is the only type which an Armv8-M processor is used to hold an SG instruction that enables software to transition from non-secure to secure state.

Non-secure (NS)

The non-secure addresses are used for memory and peripherals accessible by all software running on the device. The non-secure transactions are only permitted to access non-secure addresses.

4.8.2 Implementation defined attribution unit (IDAU)

The IDAU provides a first memory partition as non-secure or non-secure callable attributes. Its memory-map partition is not configurable, and is fixed by hardware implementation as

shown in *Table 16*. It is then combined with the results from the SAU security attribution and the higher security state is selected.

Table 16. IDAU partitioning

Address range	Security attribution
0x00000000 - 0x07FFFFFF	Non-secure
0x08000000 - 0x0BFFFFFF	Non-secure
0x0C000000 - 0x0FFFFFFF	Secure/Non-secure callable
0x10000000 - 0x17FFFFFF	Non-secure
0x18000000 - 0x1FFFFFFF	Non-secure
0x20000000 - 0x2FFFFFFF	Non-secure
0x30000000 - 0x3FFFFFFF	Secure/Non-secure callable
0x40000000 - 0x4FFFFFFF	Non-secure
0x50000000 - 0x5FFFFFFF	Secure/Non-secure callable
0x60000000 - 0xFFFFFFFF	Non-secure

4.9 Nested vectored interrupt controllers (NVIC)

4.9.1 NVIC features

The NVIC includes the following features:

- 320 maskable interrupt channels
- 16 programmable priority levels (4 bits of interrupt priority)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the Cortex-M33 core are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts.

All the interrupts including the Cortex-M33 core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to Cortex-M33 technical documentation.

4.9.2 Interrupt and exception vectors

Refer to *Section 26: Interrupt list*.

4.10 Cortex-M33 memory interfaces

Fast C-BUS

This bus connects the C-AHB interface of the Cortex-M33 core to the device interconnect via the ICACHE master1 port. This bus is used for instruction fetch and data access to the memories mapped in code region.

Slow C-BUS

This bus connects the C-AHB interface of the Cortex-M33 core to the device interconnect via the ICACHE master2 port. This bus is used for instruction fetch and data access to memories mapped in code region.

S-BUS

This bus connects the S_AHB interface of the Cortex-M33 core to the device interconnect. This bus is used for instruction fetch and data access to the memories mapped in system region, and to access peripherals. The DCACHE located on this interface is bypassed for internal memories.

The following table shows the memory connectivity with respect to the different interfaces.

Table 17. Memory partitioning(1)

Target	Fast C_BUS	Slow C_BUS	S_BUS_BYPASS	S_BUS_DCACHE
SYSRAM	P	P	P	-
SRAM1	P	-	P	-
RETRAM	P	-	P	-
FMC	-	(2)	-	P
OCTOSPI	-	(2)	-	P
DDR	-	(2)	-	P
SOC peripherals	-	-	P	-

1. This matrix shows possible memory access (P) from the Cortex-M33 core, without consideration about security filtering or memory availability.
2. This implies using remap capability of the ICACHE master2 port.

5 System security

The STM32MP21xx devices are designed with a comprehensive set of security features, some of which being built on top of the standard Arm TrustZone technology.

This section explains the different security features available on these devices.

5.1 Key security features

- Resource isolation using privilege mode and Armv8-M mainline security extension of the Cortex-M33, extended to allocated I/Os, DMA channels, memories, and peripherals
- Resource isolation using privilege mode and Arm TrustZone security extension of the Cortex-A35, extended to allocated I/Os, DMA channels, memories, and peripherals
- For other masters than the Cortex-M33 and Cortex-A35, the STM32 resource isolation framework (RIF) can allocate peripherals, fixed-sized embedded memory blocks and variable-sized address space regions, under the control of security firmware.
- Boot ROM, supporting the download of boot image through USART, USB, serial NOR or NAND flash (via OCTOSPI), parallel NAND flash (via FMC), SD-card or *e*MMC (via SDMMC)
- Non-volatile storage (in BSEC), featuring:
  - ECC protected OTP words, one-time programmable
  - Redundancy protected OTP words, bit per bit programmable
  - Embedded lifecycle management (such as sticky lock, permanent lock, tamper mode)
- Secure storage, featuring:
  - Battery-powered volatile secure storage, automatically erased in case of tamper (8-Kbyte RAM, 128-byte registers)
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID, readable via JTAG
  - Provisioning of AES wrapped keys in non-volatile storage (for example external flash memory), encrypted with unreadable device-unique keys derived from non-volatile hardware unique key (DHUK). These provisioned AES keys are never readable in the clear, are only usable on this device, and are only usable in the target AES accelerator (SAES, CRYP1, or CRYP2). If the application-defined volatile boot hardware key (BHK) is used in the wrapping, the provisioned AES keys are not usable after a tamper.
- General-purpose cryptographic acceleration:
  - Two cryptographic processors (CRYP1, CRYP2) supporting AES 256-bit and DES/triple DES with chaining modes ECB and CBC. CTR, GCM, and CCM chaining modes are also supported in AES mode.
  - Secure AES 128/192/256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel counter-measures and mitigations. Using dedicated key bus, optional key sharing with faster CRYP1 or CRYP2, controlled by the SAES.

  - Two HASH processors (HASH1, HASH2) supporting SHA-1 checksum, SHA-2 secure hash (224, 256, 384 and 512) and six Keccak-based functions (SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, and SHAKE256).
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets.
  - Two true random number generators (RNG1, RNG2), NIST SP800-90B pre-certified
- On-the-fly encryption/decryption of external DDR-SDRAM memory region(s).
  - Any secure region can be encrypted, with an automatic key erase in case of tamper, with user option to select, per region, between enhanced robustness and faster performance.
- On-the-fly decryption of encrypted image stored on external flash memory connected through the OCTOSPI:
  - Almost-zero latency with standard NOR flash memories
  - Can be used to encrypt the image using device-unique secret keys
  - Automatic key erase in case of tamper
- Secure lifecycle management (secret provisioning, return material for analysis using customer-defined 128-bit key, debug deactivation)
- Secure software monitoring using independent watchdog 3 (for Cortex-M33) and independent watchdog 1 (for Cortex-A35)
- Active tamper and protection against temperature, voltage, and frequency attacks
  - 7 tamper inputs, 5 active tamper output pins, available in all power modes
  - Immediate erase of battery-backed volatile secrets on confirmed tamper detection

5.2 Secure boot

On an application reset, the Cortex-A35 always boots from the start of an on-chip ROM. This ROM code is used to authenticate and decrypt images loaded into the internal memory from an external flash memory, or through FMC, UART, USB, OCTOSPI, or JTAG.

The lifecycle state of the device is represented in the values of specific fuses. Transforming from one lifecycle state to the next always involves blowing fuses. The BootROM acts in accordance with what it finds to be the current lifecycle state.

Lifecycle states can be partitioned into two underlying categories: BSEC-open or BSEC-closed (refer to *Section 6.3.7: Lifecycle management* for more details). This is determined by hardware on every system reset by examining the state of the lifecycle fuses.

If the hardware determines at power-up that the device is BSEC-open, it immediately enables debug, and disables any access to fused secrets and secure parts of the ROM. This is referred to as non-secure boot. The device is in a BSEC-open lifecycle state at the start of manufacturing. It can be transformed back into a BSEC-open lifecycle state if there is a fault requiring the part to be returned for analysis.

If the hardware determines that the device is BSEC-closed, the debug remains disabled. The secure boot from ROM is then enforced, and the BootROM then consults additional fuses to determine the precise lifecycle state.

Devices are delivered to customers in a BSEC-closed state, so the BootROM execution is enforced.

On a BSEC-closed device, if the secure_boot fuse bit (see *Section 7: OTP mapping (OTP)*) is still unblown, then the BootROM behavior depends on the BOOTx pin:

- If the BOOTx pins are set to "development boot", the BootROM disables the access to secrets, and opens the debug to allow customer development.
- If the BOOTx pins are not set to "Development boot", the BootROM executes the boot image regardless of whether it is correctly signed.

BOOTx pins should be set to 0011 or 1010 for "development boot" mode (see datasheet boot sources table).

However, on a BSEC-closed device, once the secure_boot fuse bit is blown, authenticated secure boot is enforced.

STMicroelectronics supplies signing tools to allow customers to create signed and encrypted boot images. The boot image is sometimes referred as an FSBL (first-stage bootloader), or as an uRoT (updatable root-of-trust firmware).

5.3 Secure provisioning

ST supplies customers with an RSSe (Root Security Services extension) code image to implement SFI (secure firmware install) functionality. The BootROM authenticates this RSSe code with a hard-coded public key. ST also provisions an ST-certified device-unique key pair into fuses. When customers receive the device, they can boot the supplied SFI image to securely provision their own keys into fuses. The BootROM uses these keys, whenever it boots, to authenticate and decrypt the customer boot images. The ST-certified device-unique key pair ensures that this provisioning process is secure and counted. This protects against cloning.

The BootROM supports up to eight irreversible revocations of firmware signing keys.

The boot image header contains a constant decrypted by the BootROM using the provisioned firmware encryption key. This derived key is used to decrypt the actual image.

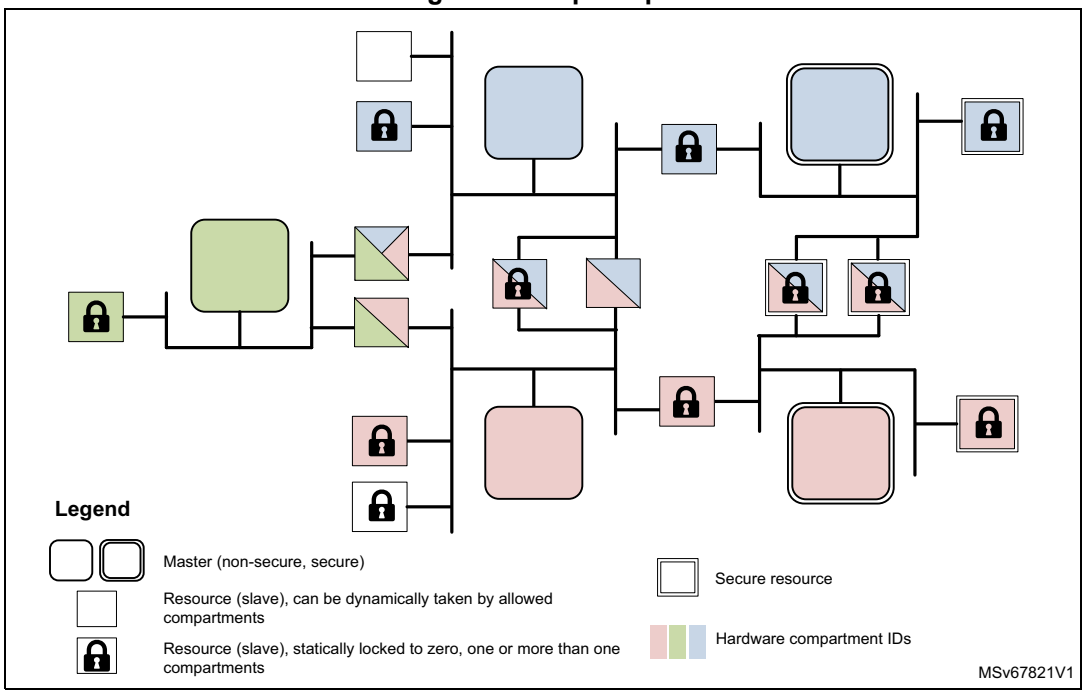
5.4 Resource isolation framework (RIF)

The resource isolation framework (RIF) is a comprehensive set of hardware blocks designed to enforce and manage the isolation of STM32MP21xx hardware resources, like memory and peripherals.

Application software can use RIF to assign hardware resources to dedicated processing environments that are isolated from each other using hardware isolation boundaries. Within these environments masters are assigned to a security domain and can be secure, non-secure, running privileged or unprivileged code, executing from a CPU, an MCU, or a complex hardware subsystem like a multimedia engine.

RIF-protected resources are assigned at boot time (and subsequently locked) or assigned at run-time. Some RIF-protected resources are exclusively assigned, some are shared, and some are unassigned, as shown in *Figure 7*.

Figure 7. RIF principle

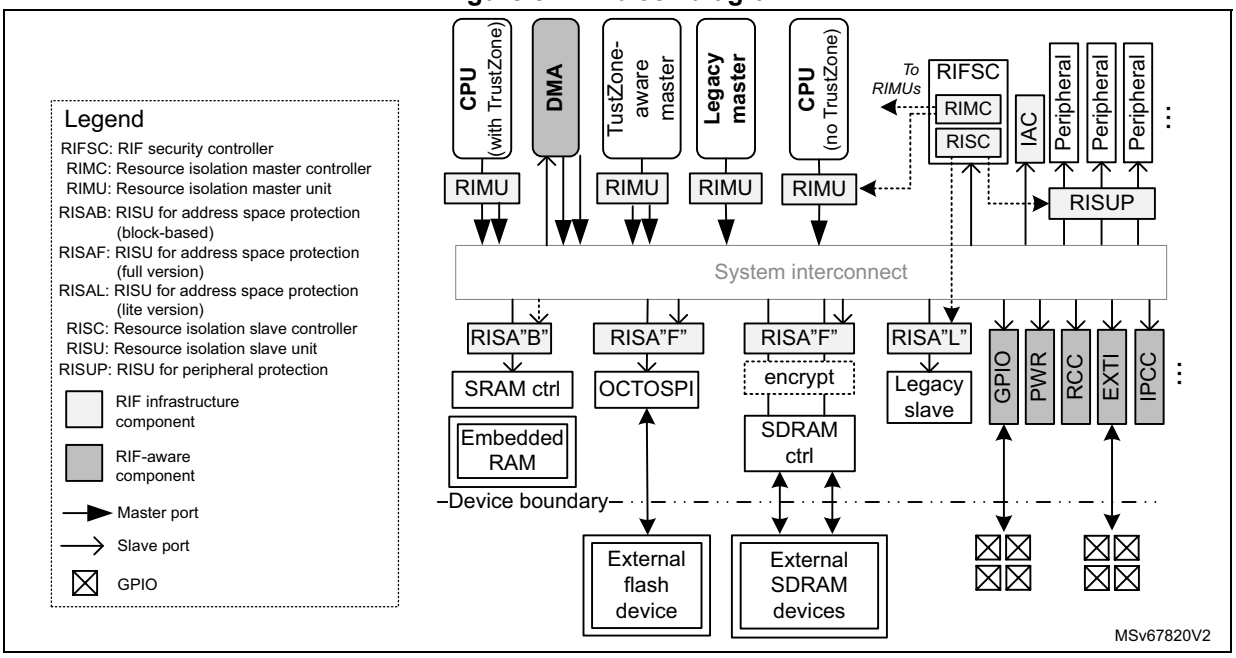


Note: In the RIF, a security domain corresponds to a triplet (privilege, secure, compartment ID).

5.4.1 RIF security architecture

The resource isolation framework (RIF) is typically implemented as described in the figure below.

Figure 8. RIF block diagram



The core components constituting the RIF are summarized below, and in *Table 18*:

- RIMU (resource isolation master unit) are components associated to masters
- RISU or RIS (resource isolation slave unit) are components associated to slave targets
  - filtering accesses to fixed-sized embedded memory blocks (RISAB), or variable sized address space regions (RISAF, RISAL)
  - filtering accesses to each peripheral (RISUP)
- The RIFSC (RIF security controller) registers contains the following blocks:
  - RIMC (resource isolation master controller) configures the RIMU components.
  - RISC (resource isolation slave controller) configures the RISUP and RISAL components.
- The IAC (illegal access controller) concentrates all illegal access events managed through the RIF framework (see *Section 5.4.9*)

When a peripheral needs a finer-grained filtering (like DMA, GPIO, or RTC), it becomes **RIF-aware**, implementing the needed resource isolation policy within itself (see *Section 5.4.8*).

Table 18. Overview of RIF core components

-	RIFSC			RISAF	RISAB
	RIMU	**RISUP**	**RISAL**		
Affecting target	Initiator	Target			
Resource type	Fixed		Variable		Fixed
	Master port	Peripheral	Address space regions		Memory block
Access filtering	N/A	Privilege, secure, compartment ID			
Read/write filtering		No		Yes	
Special feature		Semaphore	N/A	Delegation	

Refer to the following sections for more details:

- *Section 8: Resource isolation framework security controller (RIFSC)*
- *Section 9: Resource isolation slave unit for address space protection (block-based) (RISAB)*
- *Section 10: Resource isolation slave unit for address space protection (full version) (RISAF)*

As shown on *Table 18*, the access filtering can be, depending on the case:

- restricted to none, one or more than one hardware compartment IDs
- secure-only, non-secure only, or both
- privileged-only or privileged/unprivileged
- read-only, write-only, or read/write

Depending on the device, a couple of special features are available:

- in-built hardware semaphores to dynamically change ownership of resources
  - This method is also available for GPIOs, DMA channels, as well as control of power, clock, and reset (in RCC and PWR).

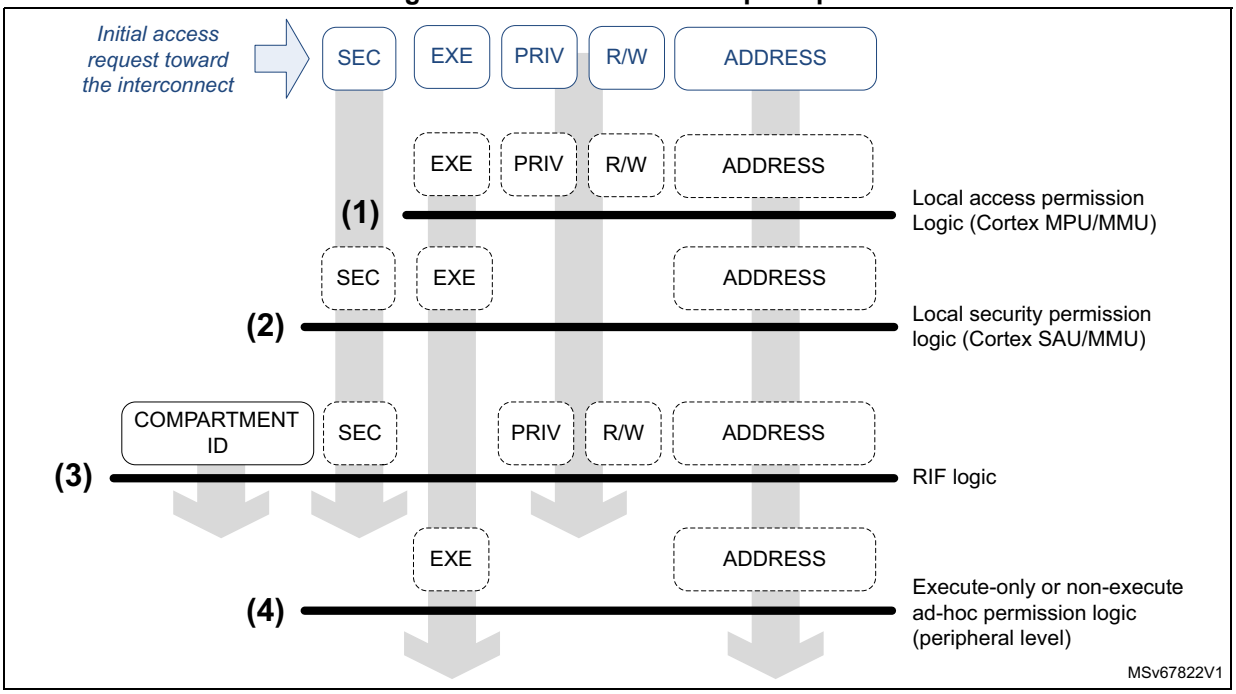
- delegation of resource isolation configuration software (from trusted domain) to the processing environment that owns the resource

Any RIF configuration of a resource can be locked at boot time.

5.4.2 RIF access control

The RIF access control principle is outlined in the figure below.

Figure 9. RIF access control principle



For Arm Cortex processors, each access to an hardware resource goes through the following steps:

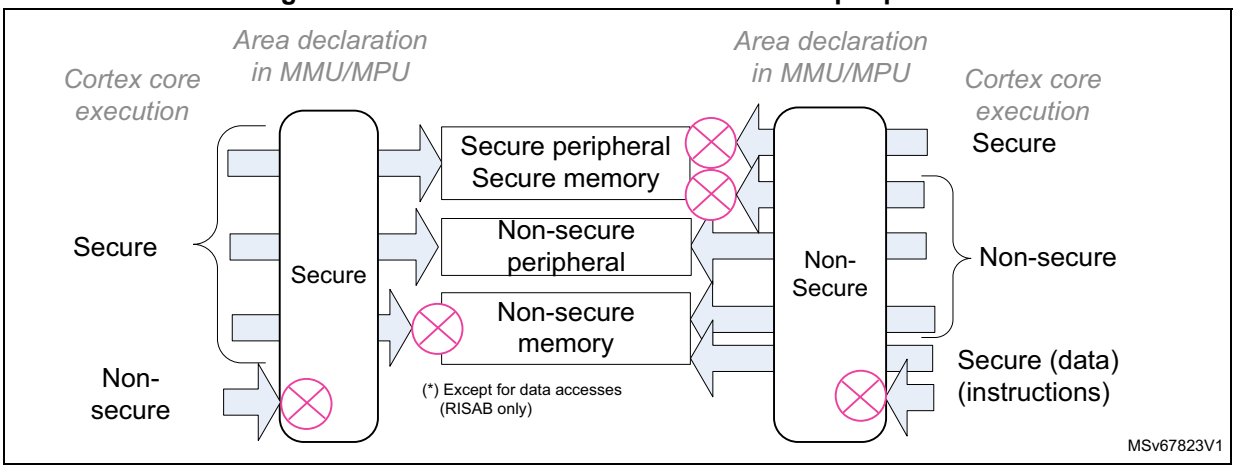
1. Within the cores, the MPU or MMU performs filtering based on privilege, execute, and read/write criteria.
2. Still within the cores (when TrustZone is supported), a secure/non-secure access control is done, sometimes taking into account whether access is instruction or data.
3. The RIF verifies that the access is granted based on the compartment ID of the access initiator. Additional controls based on secure and privileged properties are also performed. Read/write extra control is supported for some RIF resources (such as embedded memory pages).
4. Finally, some resources performs extra-filtering like no-execute on banks of registers.

Accessing secure memories and peripherals

An application executing in non-secure state can only access non-secure memories and peripherals. An application executing in secure state can access both secure and non-secure memories and peripherals.

In the latter case, there is an important rule to follow: A secure application must access a non-secure memory or peripheral through a non-secure alias in the MMU/MPU table, as explained in *Figure 10*.

Figure 10. Accesses to secure memories and peripherals



More specifically, when a secure application wants to write data to the non-secure world, the secure data access is transformed automatically by the core into a non-secure data access, as the target region is mapped into a non-secure space in the MMU (Cortex-A) or SAU/IDAU (Cortex-M).

Note: This secure to non-secure conversion is also supported by the DMA, under the supervision of a secure application.

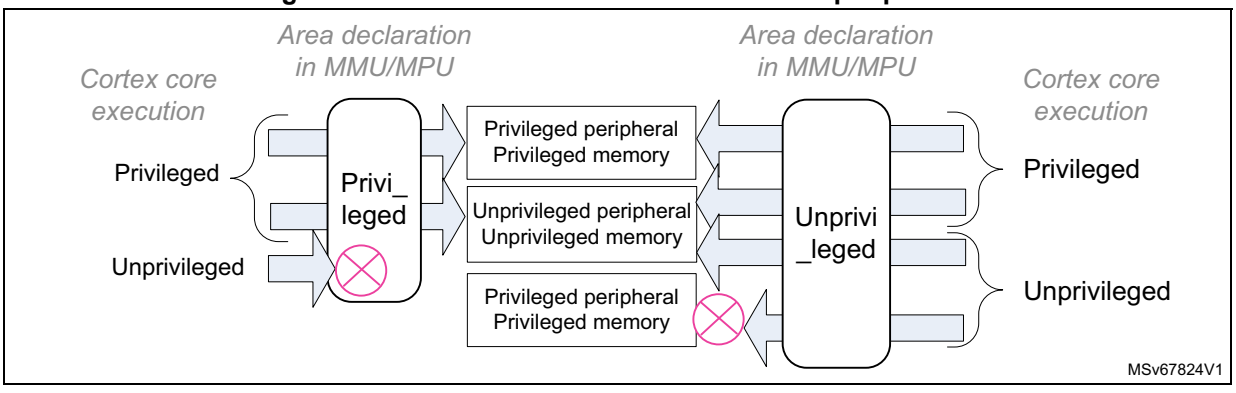
The RISAB supports the option of deactivating the secure read/write illegal access to non-secure memory (see SRWIAD bit in RISAB_CR).

Mapping peripherals in two address spaces (secure and non-secure) is optional for Cortex-A.

Accessing privileged memories and peripherals

An application executing in unprivileged state can only access unprivileged memories and peripherals. An application executing in privileged state can access both privileged and unprivileged memories and peripherals. Unlike with security, a privileged application can always access an unprivileged memory or peripheral, as shown in the figure below.

Figure 11. Accesses to secure memories and peripherals



5.4.3 Application interrupt management in RIF

As shown on *Figure 7*, a secure application can allocate or deallocate any resource to any compartment thanks to the RIF. The interrupt infrastructure between peripherals and CPU/MCUs must take into account this flexibility, with each compartment being able to see only the interrupt events of the resources allocated to its CPU/MCU.

Figure 12 and *Table 19* describe the various interrupt management options in a dual-CPU context.

Note: When the compartment filtering is disabled (CFEN = 0), the interrupts are broadcasted to all CPU/MCUs. When a peripheral is not allocated to any compartment (CFEN = 1, SEM_EN = 1, and SEM_MUTEX = 0), no CPU/MCUs receive interrupts from this peripheral.

Figure 12. RIF interrupt connection types

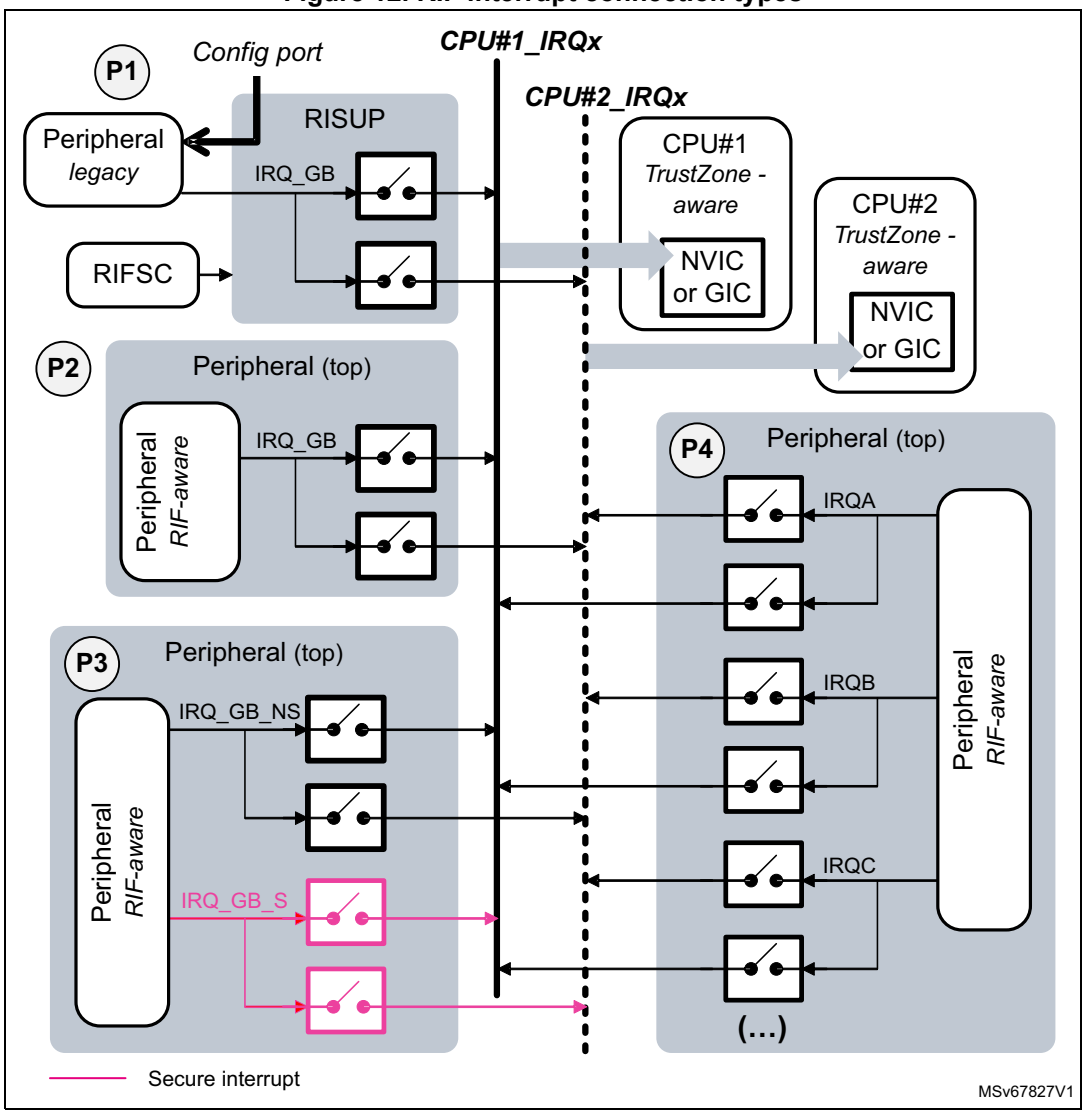


Table 19. Interrupt connection types

Peripheral type	RIF aware(1)	Global IT(2)	Secure/ non-secure IT	Interrupt events destination
P1	No	Yes	No	The allocated CPU, for all events (secure and non-secure)
P2	Yes			
P3			Yes	
P4	Yes(3)	No	No	The CPU concerned by its specific events (secure or non-secure, according to event resource setup)

1. If yes, the peripheral is RIF-aware. It manages resources that are associated to more than one CID.
2. If yes, a single interrupt line captures all the interrupt events.
3. Any interrupt status bit in this peripheral must be readable only by the correct CPU in the correct security and privileged mode.

Note: The debug domain does not manage interrupts.

5.4.4 RIF illegal access definition

When RIF access control is enforced, illegal access events are defined in following tables.

Table 20. Access-type definitions for Arm TrustZone cores

Processing environment	Core access	MPU(1)/MMU setup	Access type
Secure	Data	Secure	Secure
		Non-secure	Non-secure
	Instruction	Secure	Secure
		Non-secure	Denied(2)
Non-secure	Data	Secure	Denied(2)
		Non-secure	Non-secure
	Instruction	Secure	Denied(2)
		Non-secure	Non-secure

1. Actual configuration in Cortex-M33 is done as a mix of programmable SAU and fixed IDAU.
2. A fault is triggered within the core.

Table 21. RIF illegal access definitions (security)

Access type (see *Table 20*)		Memory resource		Peripheral resource	
		Secure	Non-secure	Secure	Non-secure
Secure	Data	OK	Silent fail(1)(2)	OK	OK
Non-secure		Silent fail(2)	OK	Silent fail(2)	OK
Secure	Instruction	OK	Bus error	Bus error	Bus error
Non-secure		Bus error	OK	Bus error	Bus error

1. Always OK for debug domain. Configurable as OK for any compartment if SRWIAD = 1 in RISAB_CR.
2. Secure illegal access event, also triggered toward the IAC.

Table 22. RIF illegal access definitions (privilege)

Access type (see *Table 20*)		Memory resource		Peripheral resource	
		Privileged	Unprivileged	Privileged	Unprivileged
Privileged	Data	OK	OK	OK	OK
Unprivileged		Silent fail(1)	OK	Silent fail(1)	OK
Privileged	Instruction	OK	OK	Bus error	Bus error
Unprivileged		Bus error	OK	Bus error	Bus error

1. Privileged illegal access event, also triggered toward the IAC.

Table 23. RIF illegal access definitions (compartments)

Access type		Memory resource		Peripheral resource	
		CID1	CID2	CID1	CID2
CID1	Data	OK	Silent fail(1)	OK	Silent fail(1)
CID2		Silent fail(1)	OK	Silent fail(1)	OK
CID1	Instruction	OK	Bus error(1)	Bus error	Bus error
CID2		Bus error(1)	OK	Bus error	Bus error

1. Compartment illegal access event, also triggered toward the IAC.

As shown in *Table 20*, for a core executing in a TrustZone secure processing environment, the memory and peripheral resources must be mapped into a secure space.

A silent fail is defined as read as zero (for data reads) and write ignored (for writes). An illegal access signaling to the IAC usually also occurs (see *Section 5.4.5*).

A bus error is defined as read as zero and generation of an error on the system bus, so that the Cortex core can get the fault address. Bus errors do not raise any error flags in the device.

Note: A read as zero (RaZ) answer to an illegal instruction fetch also generates an HardFault in Cortex cores, because when decoding all-zero instruction code, the core triggers an HardFault exception.

Events not generating an illegal access

When a valid access tries to read (resp. write) to a read-only (resp. write-only) register protected by RIF, the access is denied (RaZ/write ignored), without the generation of an illegal access.

During a register read (resp. write) access, if at least one bitfield is not read (resp. write) xx-protected, meaning that at least one bitfield can be read (resp. written), then the output signal xx_ilac is not asserted (xx = sec, priv or cid).

For legacy peripherals protected by RISUP, no compartment access error is triggered when the semaphore is taken via one of RISC_PERy_SEMCR registers, unless the compartment of this access is not part of the white-listed compartments.

5.4.5 Managing illegal accesses with RIF

The RIF components do not use the standard interrupt lines toward NVIC or GIC concentrators. The IAC, captures all RIF interrupt events, called illegal access events.

Such events are triggered in the following cases:

- illegal access to a RIF protected resource because of security
- illegal access to a RIF protected resource because of privilege
- illegal access to a RIF protected resource because of bad CID

See *Section 11: Illegal access controller (IAC)* for more information on the way illegal accesses are managed.

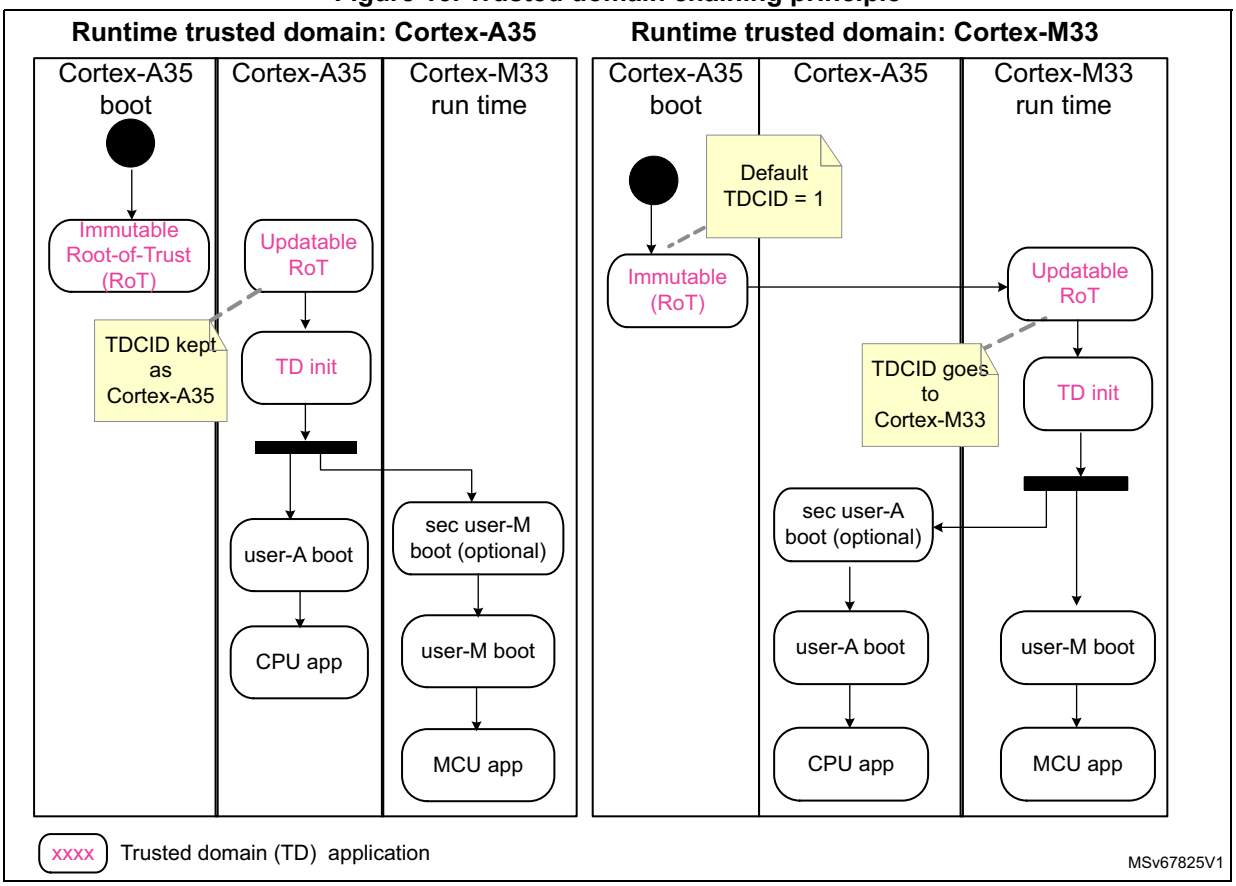
5.4.6 RIF special security domain: trusted domain

In the RIF, a master with a security domain that equals to secure, privileged, and CID = TDCID, is part of a special domain called the trusted domain. Such a master is initialized as part of the trusted boot in order to manage the critical RIF configuration.

The trusted domain compartment TDCID is fixed by default to 0x1, and is stored in RIMC_CR. The default TDCID corresponds to the boot ROM executing on Cortex-A35.

The devices support two trusted domain scenarios, shown in the figure below.

Figure 13. Trusted domain chaining principle



In the first scenario, a typical chain of trust executes on the default trusted CPU (Cortex-A35 secure privileged). In the second scenario, after securely installing MCU firmware in the embedded SRAM, the Cortex-A35 transfers the trusted domain ownership to the Cortex-M33 secure privileged, by setting TDCID = 0x2 in RIMC_CR.

Note: A TDCID transfer is possible only once.

5.4.7 RIF special security domain: debug domain

In the RIF, a privileged master with CID = DDCID, is part of a special domain called the debug domain. Such a master (usually a DAP peripheral) is initialized as part of the trusted boot to manage the debugging of a RIF protected device.

Note: In the STM32MP21xx devices, DDCID is fixed to value 0x7.

The debugging tools with debug domain access rights (privileged, DDCID) can access to all peripheral and memory resources in the device, if their security level is sufficient.

More specifically:

- Non-secure debug domain can only access peripherals and memories that are tagged by application as non-secure.
- Secure debug domain can read and write anywhere in the device, including registers reserved to the trusted domain.

Note: The debug domain master is always privileged.

When the debug domain is activated, DDEN = 1 in RIMC_CR. It is always activated in STM32MP21xx devices.

The debugging tool access right setup by trusted application is detailed in *Section 6.3.11: BSEC debug and trace control*. For more information on RIF debug access port security, see *Section 8: Resource isolation framework security controller (RIFSC)*.

5.4.8 Managing security in RIF-aware peripherals

The resources isolated by the RIF core framework defined in *Section 5.4.1* are peripherals, fixed-sized embedded memory blocks, and variable-sized address space regions.

As a RIF framework extension, additional resources are available in this device (see corresponding sections for more details).

- central DMA channels (see HPDMA)
- reset and clock control (see RCC)
- power control (see PWR)
- general purpose I/Os (see GPIO)
- real-time clock and tamper/backup registers (see RTC and TAMP)
- inter-processor communication controller (IPCC)
- flexible memory controller (FMC)
- system configuration controller (SYSCFG)
- extended interrupts and event controller (see EXTI)

5.4.9 Activating RIF security

RIF state following a system reset

After a general reset (POR, power-on reset), all non RIF-aware peripherals become non-secure and unprivileged (RISC_SECCFGRx and RISC_PRIVCFGRx are zeroed).

For embedded SRAM protected by RISAB (resp. RISAL), default values are defined in *Section 9.3: RISAB implementation* (resp. *Section 8.3: RIFSC implementation*).

Address spaces protected by RISAF are by default secure, privileged, and TDCID only.

Note: Out-of-reset compartment-based filtering is bypassed only for peripherals and for memories protected by RISAB.

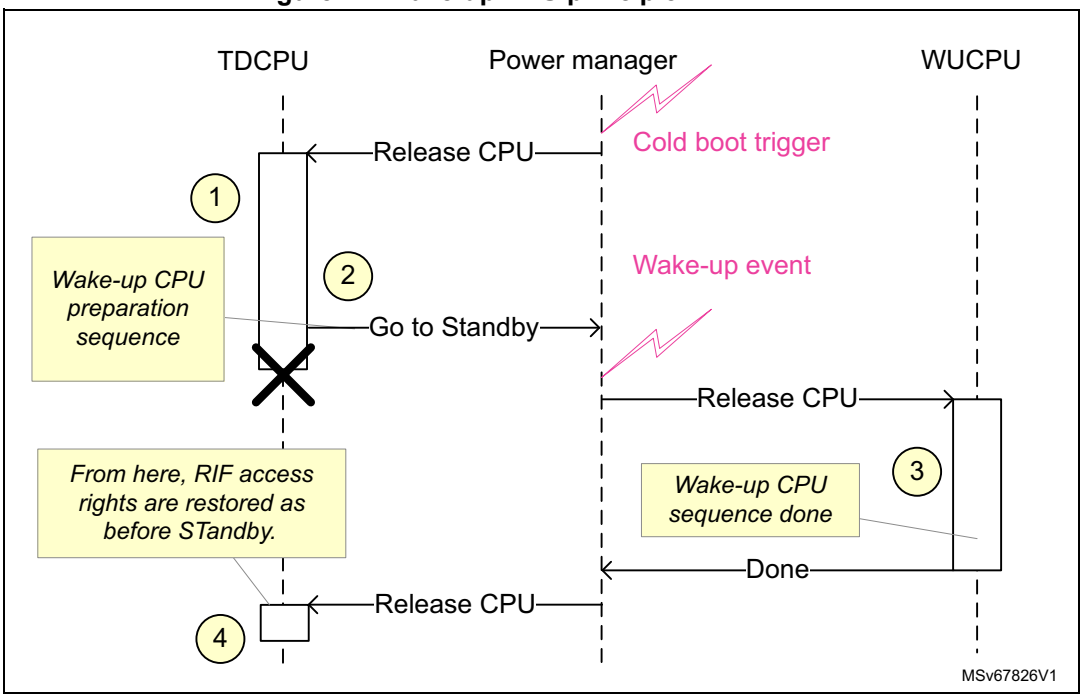
RIF wake-up from standby

When the device leaves the Standby state (see PWR), the RIF allows the Cortex-M33 to execute a RIF restoration sequence, temporarily upgrading CID = 0x2 as TDCID in RIFSC_RIMC.

Note: When the wake-up CPU option is enabled, the TDCID information out of reset from RIFSC is not RIFSC_RIMC_CR default value 0x1 (Cortex-A35), but the wake-up CID value 0x2 (Cortex-M33).

A typical wake-up CPU sequence is described in the figure below.

Figure 14. Wake-up CPU principle



The main steps are detailed below:

1. The trusted CPU executes following a cold boot trigger. The RIF is properly initialized while the applications are installed in the device.

2. The trusted CPU configures the device to allow the Cortex-M33 to quickly restore the context that is lost in Standby state. Following steps are expected:
   – Store restoration firmware in RETRAM, writing reference CRC in RAMCFG.
   – Set up wake-up event mode in the RCC, and adapt the EXTI2 configuration accordingly. The trusted domain also sets in the RCC, the necessary configuration to have wu_cpuen signal to RIFSC set.
   – Enable automatic CRC and ECC protection in RAMCFG.
3. When a wake-up event occurs, the Cortex-M33 executes restoration firmware stored in the RETRAM, if CRC is correct and if no double-error detection (DED) occurred on the RETRAM. By hardware, WUCEN is also set in RIMC_SR. At the end of the restoration sequence, the Cortex-M33 sets WUCDONE in RIMC_CR. The TDCID value goes back to the original one, stored in RIMC_CR.
4. The power manager releases the CPUs to wake up. The device is fully functional.

5.4.10 Programming of RIFSC, RCC and IAC

In order to properly program the RISUP and the RCC, and to read back illegal access information in the IAC, refer to the peripheral index tables (in *Section 8.3: RIFSC implementation*), as follows:

- When an illegal access occurs, the secure privileged application running in the trusted domain can find the source of the error using the given index x:
  – IAIEx bit to set/clear in IAC_IER
  – IAFx bit to read (resp. write) in IAC_ISR (resp. IAC_ICR)
- For legacy peripherals protected with RISUP (that is, all peripherals in *Table 38: Indexes for non-RIF-aware peripherals (RISUP)*, without footnote #2 or #3), the index x provides the following extra information:
  – SECx bit to set/clear in RISC_SECCFGR
  – PRIVx bit to set/clear in RISC_PRIVCFGR
  – RLOCKx bit to set in RISC_RCFGLOCKR
  – which RISC_PERx_CIDFGR to use: offset = 0x100 + 0x8 * x
  – which RISC_PERx_SEMCR to use: offset = 0x104 + 0x8 * x
  – which RCC register to use for clock enable and reset: offset = 0x700 + 0x4 * x
- For the local resources protected in RIF-aware RCC peripheral, the index y (RCC index) provides the following extra information:
  – SECy bit to set/clear in RCC_SECCFGR
  – PRIVy bit to set/clear in RCC_PRIVCFGR
  – RLOCKy bit to set in RCC_RCFGLOCKR
  – which RCC_RyCIDFGR to use: offset = 0x030 + 0x8 * y
  – which RCC_RySEMCR to use: offset = 0x034 + 0x8 * y

6 Boot and security control (BSEC)

6.1 Introduction

The boot and security control (BSEC) peripheral manages the accesses to an embedded one time programmable (OTP) array of fuses. Those fuses are used to store on-chip, non-volatile data like boot and security parameters.

Embedded non-volatile secrets are stored in BSEC upper area that is only accessible while BSEC is operating in its BSEC-closed state. When the BSEC state is BSEC-open, those non-volatile secrets are permanently hidden.

BSEC usage is reserved to the trusted domain CPU, but it can also be accessed by the boot CPU ROM code following a BSEC reset (cold/warm or hot). In addition, the non-trusted CPU can access its own HDPL registers.

Note: In this section, boot CPU refers to the Cortex-A.

6.2 BSEC main features

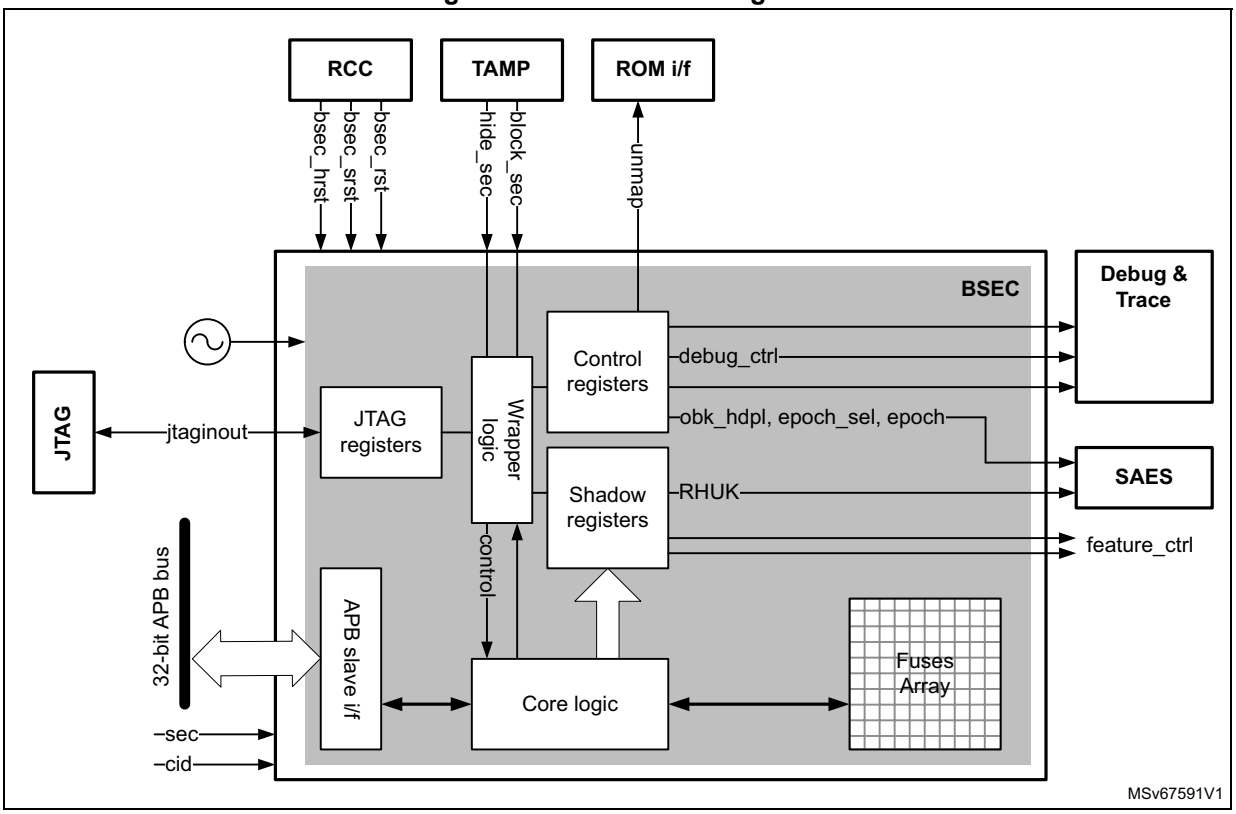
- APB peripheral, accessible through 32-bit word accesses only.
- 12 Kbits of OTP (effective)
  - 4096 lower OTP bits, bitwise (1-bit) programmable
  - 4096 middle OTP bits, bulk (32-bit) programmable
  - 4096 upper OTP bits, bulk (32-bit) programmable, only accessible when BSEC is in the BSEC-closed state.
- Reading and programming of OTP fuses, with detailed status reporting
- Shadowing in register flops of predefined fuse words, loaded at BSEC cold and warm resets. Shadow fuse registers are writable unless write locked until next BSEC cold or warm reset.
- Locking of OTP word programming, permanently or until next BSEC cold and warm resets. Option of global programming lock until next BSEC cold and warm reset.
- Read locking of fuse words (word granularity) until next BSEC cold or warm reset
  - Locked shadowed words cannot be reloaded from the fuse array (fixed value)
  - Locked unshadowed words read as 0.
- Device lifecycle management.
  - Read as 0 of upper fuse words (including RHUK) when STMicroelectronics engineering modes are enabled (BSEC-open state).
  - Possibility to go to BSEC open state using a 128-bit password (four trials).
  - Non-volatile key storage revocation counter (epoch).
- Four 32-bit scratch registers to store values persistent across device warm reboot.
- IN and OUT 32-bit registers to communicate information with external agent via JTAG.
- Multi-core debug and trace lifecycle management, with global configuration locking until next BSEC reset.
- Monotonic hide protection level counters allowing temporal isolation of keys and debug ability for each core.
- Support for TrustZone® and STM32 resource isolation framework (RIF).
- Non-volatile storage of randomly generated 256-bit root hardware unique key (RHUK), directly delivered to SAES peripheral. This key is never readable by any software, and it can be zeroed until next BSEC cold or warm reset.

6.3 BSEC functional description

6.3.1 BSEC block diagram

Figure 15 shows the BSEC block diagram.

Figure 15. BSEC block diagram



6.3.2 BSEC internal signals

Table 24 describes the user relevant internal signals interfacing the BSEC peripheral.

Table 24. BSEC internal input/output signals

Signal name	Signal type	Description
bsec_rst	Digital input	BSEC cold/warm resets from RCC (see *Section 6.3.17*).
bsec_srst		
bsec_hrst		BSEC hot reset from RCC (see *Section 6.3.17*).
jtaginout	Digital input and output	Interface to JTAG, including clock and reset.
hide_sec	Digital input	Set by TAMP in case of confirmed tamper (see *Section 6.3.17*).
block_sec	Digital input	Set by TAMP in case of potential tamper (see *Section 6.3.17*).
rhuk	Digital output	256-bit root hardware unique key output to SAES peripheral.

Table 24. BSEC internal input/output signals (continued)

Signal name	Signal type	Description
obk_hdpl	Digital output	Control outputs to SAES peripheral to influence generation of DHUK.
epoch_sel epoch	Digital output	Control outputs to SAES peripheral to influence generation of DHUK.
debug_ctrl	Digital output	Device debug and trace control signals (see *Section 6.3.11*).
feature_ctrl	Digital output	Device feature control signals (see *Section 6.3.12*)

6.3.3 BSEC reset and clocks

BSEC flops on the cold reset retain their values through warm reset, while the rest of the system remains in reset until the auto-load process has completed (*bsec_done* = 1) and fuse output values are stable.

On a POR, the BSEC temporarily emerges from warm reset for the duration of the auto-load process, even if *bsec_rst* remains asserted, so that the auto-load process can determine values for flops on the cold reset. This means when the device is powered up, the security state is immediately established, so that a debugger may be allowed to perform discovery operations soon after power-up, even if the PAD reset is still being asserted.

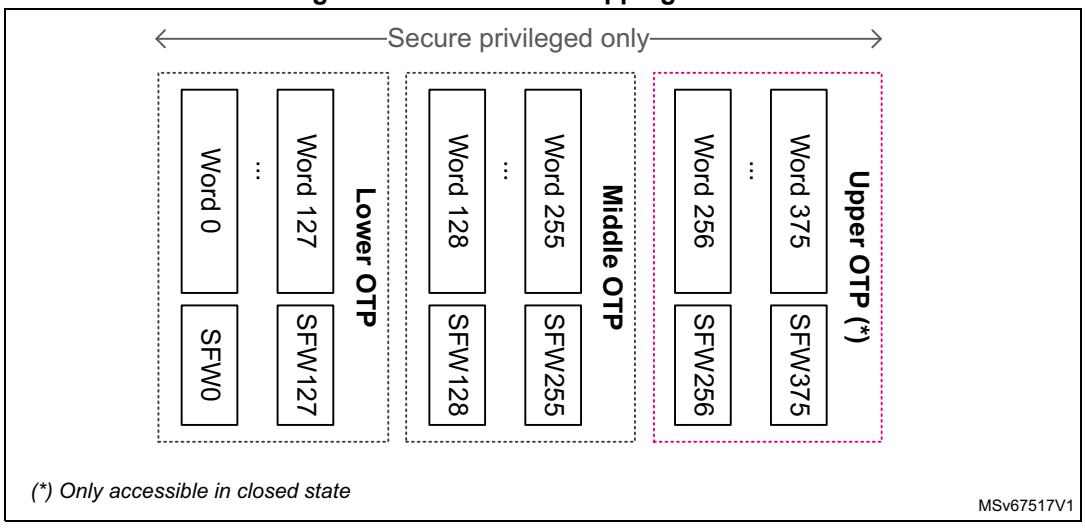
Some flops in the BSEC are on *bsec_hrst*, so they remain in reset until *bsec_done* is asserted. This reset is asserted while the Cortex-A is under reset.

6.3.4 Organization and use of fuses

Organization of fuses

One-time programmable fuse bits are organized in three groups of 32-bit words, as shown on *Figure 16*.

Figure 16. BSEC fuse mapping overview



The **lower** fuses region is used to store non-volatile, non-secret information that may require bitwise update during the life of the product.

The **middle** fuses region is used to store non-volatile, non-secret information that are written once, in 32-bit words bulk.

The **upper** fuses region is similar to the middle region, except that it is hidden when in the BSEC-open state (see *Section 6.3.7*). Hence it is used to store secrets like symmetric keys or private asymmetric keys.

Each fuse word w is either “shadowed” or “unshadowed”, as defined by flag bit SFWw in corresponding BSEC_SFSRx register. A shadowed fuse word is loaded into flops at cold or warm reset, and can be read quickly by software. An unshadowed fuse word is not read out of the fuse memory at reset, then stored permanently in flops. It needs to be reloaded from the fuse memory each time it is needed. See *Reading fuses* in *Section 6.3.5* for details.

Note: The whole OTP space can only be controlled by a secure privileged CPU.

Each individual lower fuse bit is one-time programmable, from 0 to 1. Transitions from 1 to 0 are not possible.

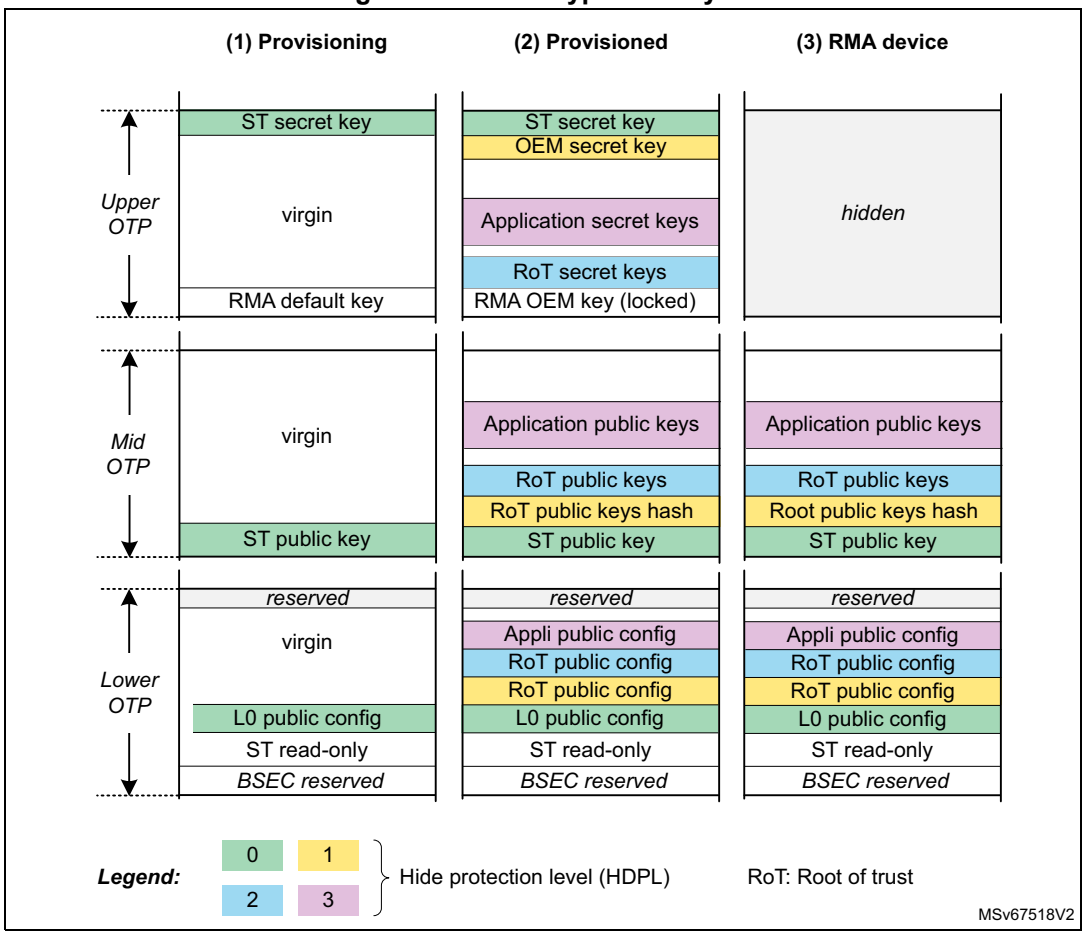
Use of fuses

Figure 17 shows how one-time programmable fuse bits are typically used during the life of a device:

1. Device being provisioned with root of trust and application keys, on an untrusted OEM production line or using STM32 tools
2. Device on the final personalization line of the OEM, or in the field
3. Return material for analysis (RMA) device, decommissioned by OEM or STMicroelectronics to investigate a particular product.

The state is BSEC-open for case (3), and BSEC-closed for cases (1) and (2). For more details on BSEC state transitions, refer to *Section 6.3.7*.

Figure 17. Device typical lifecycle



Boot levels, with associated isolation enforcements, are described in *Section 6.3.10*.

Details on fuse usage are described in *Section 7: OTP mapping (OTP)*.

6.3.5 Operations on fuses

Reading fuses

If fuse word w is shadowed (SFWw bit set in corresponding BSEC_SFSRx register), and either the state is BSEC-closed or w < 256, then BSEC_FVRw register is auto-loaded at cold or warm reset with the contents of fuse word w.

If fuse word w is accessible and shadowed, then if SRLOCKw = 0 the contents of fuse w can be reloaded by software into BSEC_FVRw register.

If fuse word w is accessible and unshadowed (SFWw bit cleared in corresponding BSEC_SFSRx register), a read to BSEC_FVRw register returns the content of fuse word w if the last operation on the fuse memory successfully reloaded word w, and if no unshadowed fuse word has been read since that reload.

Each time the application reloads the content of a word stored in fuse memory using BSEC_OTPCR register with PROG bit cleared, BUSY bit and error status should be checked in BSEC_OTPSR (see *Section 6.3.6*).

Note: Unshadowed fuse word information stored in BSEC_FVRw is cleared as soon as it is read.

Only the CPU that requested a fuse word load through BSEC_OTPCR can read the data fetched from fuses.

Writing shadow registers

When a fuse word w = {i + 32 * x} above word 9 is shadowed (SFWw bit set in BSEC_SFSRx) the application can update its shadowed value by writing directly to BSEC_FVRw register, if SWLOCKw bit is cleared in BSEC_SWLOCKx register.

Note: A shadow register write does not change the underlying fuse value.

Programming fuses

When programming is not blocked, lower fuse words can be blown bit-per-bit at different times. Middle and upper fuses must be programmed 32-bits at a time, only once. As reprogramming middle and upper fuses can let them with invalid content (an ECC error might be reported), it is strongly recommended to set PPLOCK = 1 after the first (and only) programming of the middle and upper fuses.
The procedure for blowing fuse word w is as follows:

1. Write the value to be blown in BSEC_WDR, then set ADDR = w, PROG = 1 and PPLOCK = 0 in BSEC_OTPCR register.
2. When BUSY bit is cleared in BSEC_OTPSR, verify that PROGFAIL is cleared. If PROGFAIL is set try again step 1.
3. Reload the fuse word w using BSEC_OTPCR with PROG bit cleared.
4. When BUSY bit is cleared in BSEC_OTPSR, verify that BSEC_FVRw register returns the correct value. If not, try again step 1.

Note: OTP bits are initially set to 0 and once a bit is set to 1 it cannot be programmed back to 0 (bit stays at 1). Bitfields with 0s in BSEC_WDR are not programmed, while bitfields with 1s are programmed.

As lower fuse words can be programmed multiple times, when the application requests a specific bit to be programmed to 1 more than once, BSEC does the burning only once.

Permanent program lock (PPLOCK)

The application can make further programming of fuse word w impossible by following the sequence:

1. Clear BSEC_WDR register, then set ADDR = w, PROG = 1 and PPLOCK = 1 in BSEC_OTPCR register.
2. When BUSY bit is cleared in BSEC_OTPSR, verify that PROGFAIL is cleared. If PROGFAIL is set try again step 1.
3. Reload the fuse word w using BSEC_OTPCR with PROG bit cleared.
4. When BUSY bit is cleared in BSEC_OTPSR, verify that the PPLF in BSEC_OTPSR is set. If not, try again step 1.

There is no register in BSEC to store PPLOCK information. When the application needs to know if a fuse word programming is permanently locked, it has first to perform a reload of this word using BSEC_OTPCR and then check if the PPLF bit in BSEC_OTPSR register is set. See *Table 26* for details.

Fuse word locking features

In order to make fuse usage suitable for security, for example to implement the isolation scheme detailed in *Section 6.3.10*, BSEC has the following locking features:

Sticky program lock (SPLOCK)

The application can prevent fuse word w = {i + 32* x} programming until next BSEC reset (cold or warm) by setting the SPLOCKw bit in BSEC_SPLOCKx register.

Sticky write lock (SWLOCK)

The application can make the BSEC_FVRw register of a shadowed fuse word w = {i + 32 * x} unwritable until next BSEC reset (cold or warm) by setting the SWLOCKw bit in BSEC_SWLOCKx register.

Sticky reload lock (SRLOCK)

Reload of word w = {i + 32 * x} using BSEC_OTPCR register can be blocked until next BSEC reset (cold or warm) by setting the SRLOCKw bit in BSEC_SRLOCKx register.

When read is sticky-locked, unshadowed fuse words return 0 while shadowed fuse words return their current value (value is not zeroed).

Upper fuses read and write locking (HIDEUP)

When HIDEUP bit is set by hardware in BSEC_OTPSR, neither read nor write are possible to fuse words located in the upper region.

6.3.6 BSEC read and programming status reporting

After a cold and warm reset BSEC performs a first set of OTP reads, reporting the specific status structure defined in *Table 25*.

Table 25. BSEC initial status reporting structure

Bit offset	BSEC_ OTPSR bit	Name	Error flag	Meaning when bit is set
0	[0]	BUSY	No	BSEC ongoing operation is not completed.
1	[1]	INIT_DONE		BSEC initialized following a cold or warm reset.
2	[2]	HIDEUP		Upper fuses are not accessible to the application.
4	[4]	OTPNVIR		BSEC is not virgin. This bit should always be set.
5	[5]	OTPERR	Yes	At least one error has been detected during the reading of shadowed fuse words, as reported in BSEC_OTPVLDR registers (VLDF bit = 0). The recommended mitigation is to reload the faulty fuse word using BSEC_OTPCR and analyze the 8-bit error structure in BSEC_OTPSR, as detailed below.
6	[6]	OTPSEC		At least one single-error has been corrected during the reading of shadowed fuse words, as reported in BSEC_OTPVLDR registers (VLDF bit = 0). In this case the shadowed fuse word value is correct, hence if OTPSEC = 1 and OTPERR = 0 the application does not need to reload the faulty fuse word.

After any read or programming request, the BUSY bit is cleared, and BSEC reports the specific status structure defined in *Table 26*. When this structure is different than 0 the read or program operation might have failed, as indicated in the table.

Table 26. BSEC ad-hoc error/status reporting structure

Bit offset	BSEC_ OTPSR bit	Name	Failed operation	Retry if error(1)	Error description
0	**[16]**	PROGFAIL	Write	Yes	Programming operation failed.
1	**[17]**	DISTURB	Read		Integrity of the fuse value is not guaranteed.
2	**[18]**	DEDF			Double error detected. Fuse word is corrupted.
3	**[19]**	SECF		No	Single bit error correction.
4	**[20]**	PPLF			Permanent programming lock detected.
5	**[21]**	PPLMF			Permanent programming lock information on the two lower fuse values in the array is not identical.
6	**[22]**	AMEF(2)		Yes	Fuse address mismatch. Fuse value cannot be trusted, and should be read again.

1. For reads: need to reload the fuse (reported value cannot be trusted). For writes: need to retry the programming operation.
2. Always be 0 in normal operating conditions.

6.3.7 Lifecycle management

BSEC state initialization

Following a cold or warm reset BSEC can be in the following possible security states, defined in *Table 27*.

Table 27. BSEC states definition

BSEC states(1)	Upper fuses	DHUK in SAES(2)	Full debug	BootROM access	Fuse programming
BSEC-open	Inaccessible(3)	Unusable	yes(4)	partial	Enabled
BSEC-closed	Accessible	Hardware unique	no(5)	full(6)	
BSEC-invalid	Inaccessible(3)	Unusable	no	partial	Disabled

1. Defined in NVSTATE bitfield in BSEC_SR register (0x16: BSEC-open, 0x0D: BSEC-closed).
2. Derived hardware unique key in SAES peripheral uses RHUK information from BSEC (see *Section 6.3.15*).
3. Cannot be read or programmed by any means (functional or non-functional).
4. ST engineering modes are also available.
5. Root of trust application can unlock debug features unless DENLOCK sticky bit is set in BSEC_LOCKR register. ST engineering modes are disabled, by hardware, provided bit 20 of fuse word 124 is blown.
6. At the end of its execution the boot ROM writes 0xB9D8 FF1F to BSEC_UNMAPR, reverting boot ROM access to partial.

Errors following a BSEC initialization are detailed in *Section 6.3.6*. Details about reset management can be found in *Section 6.3.17*.

When neither BSEC-open nor BSEC-closed state (NVSTATE different than 0x16 or 0x0D), the security state is considered invalid, and the most conservative security configuration is applied: that is, fuse programming is disabled, upper fuses read as 0, debug is disabled, ROM is partially unmapped and shadow registers are read only. When NVSTATE = 0x23, a confirmed tamper is active in the device (see *Section 6.3.17*).

Note: On any reset, including hot reset, if the state is BSEC-closed, the BSEC_UNMAPR register is set to 0xA1C0 DE0D, ensuring the whole boot ROM is visible.

Opening BSEC

BSEC can transition from BSEC-closed to BSEC-open only if the user enters the programmed 128-bit password via JTAG, and then performs a global reset (JTAG + device), clearing all secrets.

The recommended JTAG password programming sequence is:

1. Write the four password words to upper fuses words 256 to 259 with PPLOCK bit cleared in BSEC_OTPCR. JTAG_PSWD0 is stored in word 256, while JTAG_PSWD3 is stored in word 259.
2. Read back password value to verify it is correct
3. Write 0x0 to the same fuse words 256 to 259, with PPLOCK bit set in BSEC_OTPCR. After this, password words cannot be read nor programmed anymore by the application.

The BSEC opening sequence is described below:

1. Bring the device TAP out of reset, and enter the device secret password sequentially using the JTAG commands 0100 (SET_JTAG_PSWD0), 0101 (SET_JTAG_PSWD1), 0110 (SET_JTAG_PSWD2) then 0111 (SET_JTAG_PSWD3).
2. Trigger a device reset, without causing a TAP reset. When BSEC emerges from reset, it blows the RMA fuse word with adequate values.
3. Trigger a TAP reset to destroy any record of the entered password, and after the next cold or warm reset, the BSEC changes its NVSTATE to “BSEC-open” in BSEC_SR. It can be verified using the JTAG command 0011 (READ_SEC). BSEC returns 0x00 BSEC_DENR[19:0] & 0x1 for an BSEC-open device, 0x00 & BSEC_DENR[19:0] & 0x2 for a BSEC-closed device.
4. Ship the device to STMicroelectronics for field return analysis (RMA)

Once the device has been opened it can be closed again by blowing to 0xF the next unblown 4-bit nibble in BSEC_FVR1 (see *Closing BSEC*). At most four device re-opening attempts are allowed, regardless of whether they are successful or unsuccessful.

By blowing at least one bit in BSEC_FVR1[27:24], the application can reduce the number of re-opening cycles to only one.

If the application blows at least one bit in BSEC_FVR1[31:28] a closed device can never be re-opened.

The application can also ensure that if a closed device is ever re-opened, it is not possible to subsequently re-close it again. It is done by blowing the permanent programming lock (PPLOCK) of fuse word 1, writing 0x00000000 to BSEC_WDR then writing 0x00006001 to BSEC_OTPCR.

Note: By default BSEC opening password is 0x0. It is advised to change it.

Closing BSEC

BSEC can transition from BSEC-open to BSEC-closed by blowing some bits such that *s*[7:0] becomes greater than *r*[7:0], or *s*[7] = 1. Bytes *s*[7:0] and *r*[7:0] are defined as follows:

s[7:0] = OR(BSEC_FVR1[31:28]) | OR(BSEC_FVR1[27:24]) | ... | OR(BSEC_FVR1[3:0])

r[7:0] = AND(BSEC_FVR2[31:28]) | AND(BSEC_FVR2[27:24]) | ... | AND(BSEC_FVR2[3:0])

With the following conventions:

- “|” is concatenation operand
- OR(bits[1:0])= bit[1] OR bit[0]
- AND(bits[1:0])= bit[1] AND bit[0]

When the device is being closed, the application can limit re-opening cycles to only one attempt by blowing at least one bit in BSEC_FVR1[27:24]. To reduce re-opening cycles to zero, the application must blow at least one bit in BSEC_FVR1[31:28].

Note: For the device to become BSEC-closed, the application must perform a BSEC cold or warm reset.

6.3.8 Epoch registers

There are two 32-bit readable registers, BSEC_EPOCHR0 and BSEC_EPOCHR1, that are writable only by boot ROM accesses. It is the responsibility of the boot ROM to read bitwise fuses and populate these registers.

There is a 1-bit programmable register BSEC_EPOCHSELCR (output on *epoch_sel*) that, when programmed to *n*, ensures that BSEC_EPOCHR*n* is output on *epoch*. Both *epoch* and *epoch_sel* are wired to the SAES where they influence the generation of DHUK, derived from the RHUK.

Secure software can blow additional bits in the appropriate fuse word to increment an Epoch, thereby making impossible the use of all keys derived from the RHUK prior to that point.

6.3.9 BSEC local access control

Any access that is not both secure and privileged is blocked, with read data zeroed and write accesses ignored. Compartment IDs that are not Trusted domain CPU, debug domain, or boot CPU are also blocked.

When BSEC is used concurrently by trusted domain CPU and boot CPU, secure application should always verify that an *unshadowed* word reading as zero is a valid value. Indeed, the returned BSEC_FVR value is correct only if LASTCID value in BSEC_OTPCR register is correct, for example LASTCID = TDCID value in *RIMC_CR* for the trusted domain CPU.

When the state is BSEC-closed, password fuse words 256 to 259 reads as zero when permanent programming lock is set for those words.

6.3.10 Temporal isolation in BSEC

In order to enforce isolation between boot stages the running application in each boot level must hide the secrets it was using before proceeding to the next level of code in the boot chain. Such protections must remain until the next boot sequence.

Four boot levels, with its corresponding hide protection level (HDPL), are defined:

- Level 0 (HDPL0) corresponds to hardware state machine or ROM code execution.
- Level 1 (HDPL1) corresponds to the execution of STMicroelectronics signed code or ROMed code. This code is the first stage of the secure boot.
- Level 2 (HDPL2) corresponds to the execution of the OEM signed code, stored in embedded SRAM. This code is the second stage of the secure boot.
- Level 3 (HDPL3) corresponds to the application run-time.

Note: The first two levels are always executed by the boot CPU.

Each CPU has its own independent notion of HDPL level. Secure software running on a CPU decides when to advance its HDPL level, but the trusted CPU can also increment the HDPL level of the non-trusted CPU. The Cortex-A reverts to HDPL0 on a hot reset, and the Cortex-M reverts to HDPL0 on a warm reset.

To help enforcing such isolation between boot stages, the ROM code and the application secure boot can use the following temporal isolation hardware capabilities of BSEC:

- When BSEC is in a closed state, ROM partially hides itself at the end of its execution in level 0 or level 1, writing 0xB9D8 FF1F to BSEC_UNMAPR register.
- When BSEC is in a closed state, fuse words 356 to 375 are reloadable only when BSEC_UNMAP=0xA1C0DE0D and when the access is from Cortex-A.
- Any 32-bit OTP word can be made non-readable and/or non-writable until next cold or warm reset. It can also be made non-programmable, permanently. See Fuse word locking features and Permanent program lock (PPLOCK) methods in *Section 6.3.5*.

Note: Details on fuse usage are described in Section 7: OTP mapping (OTP).

The registers BSEC_HDPLMSR and BSEC_HDPLASR determine the current hide protection level of Cortex-M and Cortex-A respectively. The encoding offers physical robustness.

Table 28. Current HDPL coding

HDPL field	Level
0xB4	0
0x51	1
0x8A	2
0x6F	3
others	undefined

The BSEC_HDPLMSR register is reset to 0xB4 by *bsec_rst*. The BSEC_HDPLASR register is reset to 0xB4 by *bsec_hrst*. These registers are not directly writable. If less than 3, the current level in each can be incremented by writing a value of 0x60B166E7 to BSEC_HDPLMCR or BSEC_HDPLACR respectively.

The 2-bit INCR field of the BSEC_NEXTHDPLMCR (respectively BSEC_NEXTHDPLACR) register is added to the level given by BSEC_HDPLMSR (respectively BSEC_HDPLASR), but clamped at a ceiling of 3. The result is output on *obk_hdpl[7:0],* provided the SAES currently belongs to that particular CPU. If the SAES is currently not owned by either CPU, the *obk_hdpl[7:0]* output is set to 0x00.

Table 29. BSEC Next HDPL register usage

obk_hdpl [7:0]		INCR field			
		0	1	2	3
BSEC_HDPL*SR[HDPL]	0 (0xB4)	0xB4	0x51	0x8A	0x6F
	1 (0x51)	0x51	0x8A	0x6F	
	2 (0x8A)	0x8A	0x6F		
	3 (0x6F)	0x6F			
	others (0x*PQ*)	0x*PQ*			

The *obk_hdpl* output goes to the SAES where it influences the generation of DHUK, derived from the RHUK. This mechanism allows keys for a later temporal level to be generated in advance.

Cortex-M secure privileged code can always read BSEC_HDPLMSR and can always write BSEC_HDPLMCR and BSEC_NEXTHDPLMCR, even if the Cortex-M is not the trusted domain. Cortex-A secure privileged code can always read BSEC_HDPLASR and can always write BSEC_HDPLACR and BSEC_NEXTHDPLACR, even if the Cortex-A is not the trusted domain.

6.3.11 BSEC debug and trace control

Secure privileged application can select the debug and trace profiles defined on *Table 30*, writing to BSEC_DENR register. For each profile there are two proposed configuration to BSEC_DENR register:

- A *trace-only* session allows the debugger or the self-hosted application to configure components that can be used to trace events from application code without modifying this code. This option includes dumps to DDR-SDRAM via ETR.
- A *full debug* session allows the debugger to configure components that are used to set breakpoints to the application under debug, as well as configuring any available trace features.

To increase robustness against attacks, when something other than 0xDEB followed by the Hamming code of the bottom 12-bits is written to bits [31:12], the entire BSEC_DENR register is zeroed. Write to BSEC_DENR register is ignored until next BSEC cold or warm reset if DENLOCK or GWLOCK bit is set in BSEC_LOCKR register

The secure privileged application can decide to delay the opening of debug on a CPU until that CPU has reached a particular HDPL level. This condition is given in the penultimate column of the table below, and is explained after the table.

Table 30. Debug and trace profiles

Debug profile			Cortex-A35 debug	Cortex-M33 debug	HDPL condition	Value in BSEC_DENR (LSB)
Micro-controller debug	Secure(1)	Trace-only	No	Yes	dbgs_m	0xDEB3BA98
		Full debug				0xDEB3EF98
	Nonsecure	Trace-only			dbg_m	0xDEB18298
		Full debug				0xDEB02398

Table 30. Debug and trace profiles (continued)

Debug profile			Cortex-A35 debug	Cortex-M33 debug	HDPL condition	Value in BSEC_DENR (LSB)
Application processor debug	Secure	Trace-only	Yes	No	dbgs_a	0xDEB1F0DC
		Full debug				0xDEB010FE
	Nonsecure	Trace-only			dbg_a	0xDEB0909C
		Full debug				0xDEB0209E
Global debug (MCUs + CPU)	Global secure	Trace-only	Yes	Yes	dbgs_a, dbgs_m	0xDEB20ADC
		Full debug				0xDEB3BFFE
	Nonsecure, with Cortex-M33 secure debug enabled(2)	Trace-only			dbg_a, dbgs_m	0xDEB36A9C
		Full debug				0xDEB38F9E
	Nonsecure	Trace-only			dbg_a, dbs_m	0xDEB1529C
		Full debug				0xDEB0439E

1. A secure profile can debug code running in secure and nonsecure modes.
2. Secure application running on application processor cannot be debugged nor traced. Cortex-M33 secure is running protected 3rd party applications, usually not used for security.

The wrapper contains BSEC_DBGMCR and BSEC_DBGACR registers, both reset to zero on the warm reset, and both writeable only once per warm reset. The bytes of these registers are named as follows

Table 31. BSEC debug register legal values

BSEC_DBGMCR BSEC_DBGACR	Bitfield name	Legal values
31:24	AUTH_SEC	0x00, 0xB4
23:16	AUTH_HDPL	0x00, 0xB4,0x51,0x8A,0x6F
15:8	UNLOCK	0x00, 0xB4
7:0	reserved	-

The AUTH_HDPL field encodes levels in the same way as the HDPL field of the BSEC_HDPLMSR and BSEC_HDPLASR registers.

Any attempt to write a non-legal byte value into these fields results in a 0x00 being written in that byte field.

The register is forced to zero (writes have no effect) whenever the device is BSEC-open.

On a BSEC-closed device, debug is only possible on Cortex-M if BSEC_DBGMCR[UNLOCK] = 0xB4, and the HDPL level indicated in BSEC_HDPLMSR is non-zero and reaches or surpasses that indicated by BSEC_DBGMCR[AUTH_HDPL]. In this case, secure debug is only possible if BSEC_DBGMCR[AUTH_SEC] = 0xB4. These two conditions are represented by the signals *dbg_m* and *dbgs_m* respectively in *Table 30*. They gate the appropriate CoreSight™ enables.

Similarly, on a BSEC-closed device, debug is only possible on Cortex-A if BSEC_DBGACR[UNLOCK] = 0xB4, and the HDPL level indicated in BSEC_HDPLASR is non-zero and reaches or surpasses that indicated by BSEC_DBGACR[AUTH_HDPL]. In this case, secure debug is only possible if BSEC_DBGACR[AUTH_SEC] = 0xB4. These two conditions are represented by the signals *dbg_a* and *dbgs_a* respectively in *Table 30*. They gate the appropriate CoreSight enables.

The BSEC_AP_UNLOCK register is forced to zero (writes have no effect) whenever the state is BSEC-open. The *ap_unlocked* output, when taking the value 0xB4, indicates that the debugger can access the Cortex-M33 debug subsystem. It is set to 0xB4 whenever the state is BSEC-open, otherwise it is directly driven by the BSEC_AP_UNLOCK register, which is reset to 0 on a scratch reset.

A rising edge on *hide_sec* (a confirmed tamper) resets these registers to zero, and closes debug. They are not writeable until the next cold or warm reset.

6.3.12 Device features control

Device features are controlled through read-only shadowed fuse word 9.See *Section 7: OTP mapping (OTP)* for details.

6.3.13 Scratch registers

The four BSEC_SCRATCHRx registers are only reset by a cold boot. They are therefore useful to store values that must remain persistent for the application across warm boots.

The eight BSEC_WOSCRx registers are only reset by a cold or warm boot. They are therefore useful to store values that must remain persistent for the application across hot boots.

Note: BSEC_WOSCRx registers are write once.

6.3.14 JTAG registers

The BSEC_JTAGINR and BSEC_JTAGOUTR registers are available to facilitate communication between the application (for example ROM) and the debug tools connected through the JTAG interface.

6.3.15 STM32 root hardware unique key (RHUK)

BSEC stores a root HUK that has been programmed by STMicroelectronics during the device manufacturing. A successful provisioning is indicated with HVALID bit set in BSEC_SR register.

The value of this key is obtained from the RNG peripheral of the device, without disclosure to the outside word. This key cannot be read or programmed by any secure application nor debugger.

When BSEC detects an integrity issue while reading the key information from non-volatile memory, or when application sets the sticky HKLOCK bit in BSEC_LOCKR, the key is indicated as invalid. When the key is invalid the SAES peripheral uses a default fixed value.

6.3.16 BSEC error management

Shadowed fuse word automatic readout errors

After a cold or warm reset, shadowed fuse words are automatically loaded into BSEC_FVR registers. At the end of this process, when INIT_DONE is set in BSEC_OTPSR, additional error information can be found in this register, as described in *Section 6.3.6*.

Individual fuse loading errors

Upon each fuse loading from OTP memory array, an 8-bit error structure in BSEC_OTPSR register is updated to tell the application if there was a problem with this particular fuse load. This structure is defined in *Section 6.3.6*.

6.3.17 BSEC tamper response mechanisms

This device uses tamper-detection and response mechanisms that can make it immediately inoperable, resulting in the automatic and immediate erasure of any sensitive data that may be stored in the device. The device also supports the concept of potential tamper event, where the device secrets are blocked while application software verifies that such tamper event is confirmed or not.

BSEC manages above mechanisms in the following way:

- When a tamper source triggers a potential tamper event in TAMP peripheral (NOERASE = 1), BSEC masks the RHUK value to SAES, and upper fuses read as 0 (unless BSEC_UNMAP = 0xA1C0 DE0D, meaning that the boot CPU is executing).
- When a tamper source triggers a confirmed tamper event in TAMP peripheral (NOERASE = 0), or a potential tamper becomes a confirmed tamper, fuses cannot be blown or reloaded, upper fuses read as 0, BSEC_DENR register is cleared, BSEC_DBGMCR, BSEC_DBGACR, and BSEC_AP_UNLOCK registers are cleared, RHUK information is physically cleared to 0 & invalidated, and NVSTATE is set to 0x23 in BSEC_SR register. These limitations are deactivated after device secrets have been erased, and a SoC reset has been applied.

Above limitations are deactivated when the application clears the active tamper source(s) in TAMP peripheral. See *Section 64: Tamper and backup registers (TAMP)* for details.

Note: While tamper events are active the boot CPU can always execute the application firmware following an application reset or an exit from Standby for the boot CPU.

6.4 BSEC interrupts

BSEC does not support any interrupt.

6.5 BSEC registers

Those registers are accessible through the register interface of the BSEC peripheral.

If security and privileged policy allows it, when the debug domain feature is activated, the debug domain traffic goes through, regardless of any CID filtering policy.

6.5.1 BSEC fuse word w value register (BSEC_FVRw)

Address offset: 0x000 + 0x4 * w, (w = 0 to 375)

Reset value: 0xXXXX XXXX

Note: The reset value is 0x0000 0000 if the fuse word is unshadowed, defined by the fuse value if the fuse word is shadowed.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

See the bitfield description for further details on fuse word accessibility.

If fuse word is not shadowed, only the CPU that reloads fuse information using BSEC_OTPCR register can read the actual bit values (the register returns 0 otherwise). See LASTCID in BSEC_OTPCR for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FV[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FV[31:0]**: fuse value

If fuse word {w} is shadowed (SFW{w} bit set in corresponding BSEC_SFSRx register), and either the state is BSEC-closed or {w} < 256, then this register is auto-loaded at cold or warm reset with the contents of fuse word w.

If fuse word {w} is accessible and shadowed, then:

– if SRLOCK{w} = 0, fuse word {w} can be reloaded by software into this register

– if SWLOCK{w} = 0, and {w} > 9, this register can be written to (otherwise, writes are ignored).

If fuse word {w} is accessible and unshadowed (SFW{w} bit cleared in corresponding BSEC_SFSRx register), writes are ignored and provided:

– the last operation on the fuse memory successfully reloaded word {w}

– no unshadowed fuse words have been read since that reload

then a read to this register returns the content of fuse word {w} (otherwise it returns 0).

If the fuse word {w} is inaccessible, then writes are ignored and reads return 0.

Unshadowed fuse word information stored in this register is cleared as soon as it is read.

When the state is BSEC-closed:

– Fuse words 356 to 375 are accessible only when BSEC_UNMAP=0xA1C0DE0D and when the access is from boot CPU.

– Fuse words 256 to 259 read as zero when permanent programming lock is set for those words.

6.5.2 BSEC sticky programming lock register x (BSEC_SPLOCKx)

Address offset: 0x800 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SPLOCK{i + 32 * x}**: Sticky programming lock for word {i + 32 * x} (i = 0 to 31)

Setting this bit prevents permanent programming for the fuse word {i + 32 * x} until next cold or warm reset.

0: Fuse word {i + 32 * x} can be burnt in fuse memory array

1: Attempt to program fuse word {i + 32 * x} in OTP memory array is silently ignored

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

6.5.3 BSEC sticky write lock register x (BSEC_SWLOCKx)

Address offset: 0x840 +0x4 * x (x=0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SWLOCK{i + 32 * x}**: sticky write lock for shadow register {i + 32 * x} (i = 0 to 31)

When fuse word {i + 32 * x} is shadowed, setting this bit prevents the writing of the shadow register of this fuse word until next cold or warm reset.

0: Write to shadow register BSEC_FVR{i + 32 * x} is allowed

1: Writes to shadow register BSEC_FVR{i + 32 * x} are silently ignored

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

6.5.4 BSEC sticky reload lock register x (BSEC_SRLOCKx)

Address offset: 0x880 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SRLOCK{i + 32 * x}**: sticky reload lock for fuse word {i + 32 * x} (i = 0 to 31)

Setting this bit locks reloading of fuse word {i + 32 * x} from the OTP memory array until next cold or warm reset.

0: Fuse word {i + 32 * x} loading through BSEC_OTPCR is authorized.

1: Fuse word {i + 32 * x} loading through BSEC_OTPCR is denied until next cold or warm reset.

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

6.5.5 BSEC OTP valid register x (BSEC_OTPVLDRx)

Address offset: 0x8C0 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VLDF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLDF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **VLDF{i + 32 * x}**: Valid flag for shadow register {i + 32 * x} (i = 0 to 31)

This bit represents the validity of the last reload of fuse word {i + 32 * x}.

0: An error occurred while fuse word {i + 32 * x} was last reloaded. The value read from BSEC_FVR{i + 32 * x} register cannot be relied on.

1: Last reload of fuse word {i + 32 * x} was done without error.

Each VLDF bit is updated when BSEC or the allowed application reloads corresponding fuse word from the OTP array.

6.5.6 BSEC shadowed fuses status register x (BSEC_SFSRx)

Address offset: 0x940 + 0x4 * x (x = 0 to 11)

Reset value: 0x0001 EBFF (SFSR0)

Reset value: 0xFF00 000F (SFSR3)

Reset value: 0x0000 000F (SFSR8)

Reset value: 0xFF00 0000 (SFSR11)

Reset value: 0x0000 0000 (others)

Secure privileged read-only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SFW{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFW{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SFW{i + 32 * x}**: Shadowed fuse word {i + 32 * x} (i = 0 to 31)

This bit indicates whether fuse word {i + 32 * x} is shadowed or not.

0: Fuse word {i + 32 * x} is not shadowed. Fuse value must be reloaded using BSEC_OTPCR before reading BSEC_FVR{i + 32 * x} register.

1: Fuse word is shadowed in BSEC_FVR{i + 32 * x} register. Reloading the fuse value using BSEC_OTPCR is only required if the fuse value changed since the last BSEC cold or warm reset.

6.5.7 BSEC OTP control register (BSEC_OTPCR)

Address offset: 0xC04

Reset value: 0x0000 0000

Writing to this register causes an access to be made to the fuse memory.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LASTCID[2:0]			Res.	Res.	Res.
										r	r	r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PP LOCK	PROG	Res.	Res.	Res.	Res.	ADDR[8:0]								
	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **LASTCID[2:0]:** Last CID

This read-only bitfield returns the compartment ID of the CPU that performed the last write in this register.

Bits 18:15 Reserved, must be kept at reset value.

Bit 14 **PPLOCK:** Permanent programming lock

When this bit is set together with PROG bit, no further write to this fuse word is possible.
0: Fuse word at address ADDR[8:0] is programmed normally.
1: Fuse word at address ADDR[8:0] is locked, preventing further modifications.
Permanent programming lock mechanism is immutable, as it is committed in the fuse memory.

Bit 13 **PROG:** Fuse word programming

This bit determines whether BSEC operation is a read or a write. If a write is selected, the data stored in BSEC_WDR is programmed in the fuse word indexed by ADDR bitfield.
0: Fuse word read operation is required
1: Fuse word programming operation is required
While BUSY is set in BSEC_OTPSR, writes to PROG bit have no effect.

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **ADDR[8:0]:** Fuse word address

This bitfield represents the address of the fuse word in the OTP memory array.
When a lower fuse address is selected two fuse words are addressed by BSEC.
When fuse words 376 to 383 are selected, read or programming requests are denied.
When fuse words 364 to 375 are selected, programming requests are ignored.

6.5.8 BSEC write data register (BSEC_WDR)

Address offset: 0xC08

Reset value: 0x0000 0000

Secure privileged write only. Any reads return 0. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRDATA[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRDATA[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **WRDATA[31:0]:** OTP write data

The value written to this register can be written to fuse memory. This happens when the BSEC_OTPCR register is written with the PROG bit set.
When a lower fuse address is selected in BSEC_OTPCR (ADDR=0x0 to 0x3F), two fuse words are programmed by BSEC (one fuse word is programmed otherwise).
While BUSY is set in BSEC_OTPSR, writes to this register have no effect.

6.5.9 BSEC scratch register x (BSEC_SCRATCHRx)

Address offset: 0xE00 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is reset when *bsec_srst* is asserted.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SDATA[31:0]**: Scratch data

This register can be used to store values that needs to be persistent across warm reset.

6.5.10 BSEC lock register (BSEC_LOCKR)

Address offset: 0xE10

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HK LOCK	DEN LOCK	GW LOCK
													rs	rs	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HKLOCK:** Hardware key lock

0: Derived hardware unique key (DHUK) in SAES peripheral is usable if HVALID is set in BSEC_SR

1: Derived hardware unique key (DHUK) in SAES peripheral is not usable.

Once set, this bit is cleared only by BSEC cold or warm reset.

Bit 1 **DENLOCK:** Debug enable register sticky lock

0: Writes to BSEC_DENR register are allowed

1: Writes to BSEC_DENR register are ignored

Once set, this bit is cleared only by BSEC cold or warm reset.

Bit 0 **GWLOCK:** Global write lock

0: Writes to BSEC registers are allowed

1: Writes to BSEC registers are ignored (fuse programming and fuse reload disabled)

Once set, this bit is cleared only by BSEC cold or warm reset.

6.5.11 BSEC JTAG input register (BSEC_JTAGINR)

Address offset: 0xE14

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATAIN[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATAIN[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATAIN[31:0]**: JTAG input data

Reads give the last input value from the JTAG interface

6.5.12 BSEC JTAG output register (BSEC_JTAGOUTR)

Address offset: 0xE18

Reset value: 0x0000 0000

Secure privileged write only (write ignored otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATAOUT[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATAOUT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **JDATAOUT[31:0]**: JTAG output data

Application can write any value in this field to be output on the JTAG interface.

6.5.13 BSEC debug enable register (BSEC_DENR)

Address offset: 0xE20

Reset value: 0xDEB3 CFFF (BSEC-open)

Reset value: 0x0000 0000 (not BSEC-open)

When something other than 0xDEB is written to bits [31:20], the entire register is zeroed. When the correct SECDED hamming code of bits [11:0] is not written to bits [17:12], then BSEC_DENR[19:0] is zeroed.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PATTERN[11:0]												Res.	Res.	R CODE5	R CODE4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R CODE3	R CODE2	R CODE1	R CODE0	SPNID ENM	SPIDE NM	NIDEN M	DBGE NM	DBGS WEN	SPNID ENA	SPIDE NA	HDPEN	DEVIC EEN	NIDEN A	DBGE NA	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:20 **PATTERN[11:0]**: Pattern to validate write data value
If this is written with anything other than 0xDEB, then BSEC_DENR is zeroed.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **RCODE5**: SECDED code
Set this to what is being written to bit 11, otherwise BSEC_DENR[19:0] is zeroed.

Bit 16 **RCODE4**: SECDED code
Set this to the XOR of what is being written to bits 10, 9, 8, 7, 6, 5, 4, otherwise BSEC_DENR[19:0] is zeroed.

Bit 15 **RCODE3**: SECDED code
Set this to the XOR of what is being written to bits 10, 9, 8, 7, 3, 2, 1, otherwise BSEC_DENR[19:0] is zeroed.

Bit 14 **RCODE2**: SECDED code
Set this to the XOR of what is being written to bits 10, 9, 6, 5, 3, 2, 0, otherwise BSEC_DENR[19:0] is zeroed.

Bit 13 **RCODE1**: SECDED code
Set this to the XOR of what is being written to bits 11, 10, 8, 6, 4, 3, 1, 0, otherwise BSEC_DENR[19:0] is zeroed.

Bit 12 **RCODE0**: SECDED code
Set this to the XOR of what is being written to bits 17, 16, 15, 14, 13, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, otherwise BSEC_DENR[19:0] is zeroed.

Bit 11 **SPNIDENM:** Secure privilege non-invasive debug enable for Cortex-M
0: Secure privilege non-invasive debug is disabled for Cortex-M
1: Secure privilege non-invasive debug is enabled for Cortex-M

Bit 10 **SPIDENM:** Secure privilege invasive debug enable for Cortex-M
0: Secure privilege invasive debug is disabled for Cortex-M
1: Secure privilege invasive debug is enabled for Cortex-M

Bit 9 **NIDENM**: Non invasive debug enable for Cortex-M
0: Non invasive debug disabled for Cortex-M
1: Non invasive debug enabled for Cortex-M

Bit 8 **DBGENM**: Debug enable for Cortex-M
0: Disabled
1: Enabled

Bit 7 **DBGSWEN**: Self-hosted debug enable
This bit controls application driven, self-hosted debug capability of the device.
0: Software access to all debug components is disabled
1: Software access to all debug components is enabled

Bit 6 **SPNIDENA:** Secure privilege non-invasive debug enable for Cortex-A
0: Secure privilege non-invasive debug is disabled for Cortex-A
1: Secure privilege non-invasive debug is enabled for Cortex-A

Bit 5 **SPIDENA:** Secure privilege invasive debug enable for Cortex-A
0: Secure privilege invasive debug is disabled for Cortex-A
1: Secure privilege invasive debug is enabled for Cortex-A

Bit 4 **HDPEN:** Hardware debug port enable
This bit controls the hardware debug port (HDP), allowing the observation of internal signals.
0: Hardware debug tracing port is disabled
1: Hardware debug tracing port is enabled

Bit 3 **DEVICEEN:** Device debug enable
This bit controls the access to debug components via external debug port.
0: External debugger is not usable with the device
1: External debugger is usable with the device, depending on the value stored in this register.

Bit 2 **NIDENA**: Non invasive debug enable for Cortex-A
0: Non invasive debug disabled for Cortex-A
1: Non invasive debug enabled for Cortex-A

Bit 1 **DBGENA**: Debug enable for Cortex-A
0: Disabled
1: Enabled

Bit 0 Reserved, must be kept at reset value.

6.5.14 BSEC unmap register (BSEC_UNMAPR)

Address offset: 0xE24

Reset value: 0xXXXX XXXX

Note: The reset value is 0xA1C0 DE0D on a BSEC-closed device, 0xFFFF FFFF on a BSEC-open device.

Register bits cannot be cleared by application, they can only be set to 1.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only BootROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNMAP[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNMAP[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **UNMAP[31:0]**: Unmap key
If this bitfield equals 0xA1C0 DE0D, the whole ROM is visible, else ROM is partially hidden.

6.5.15 BSEC status register (BSEC_SR)

Address offset: 0xE40

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NVSTATE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HVALID	Res.
														r	

Bits 31:26 **NVSTATE[5:0]**: Non-volatile state
0x16: BSEC-open state
0x0D: BSEC-closed state
0x7: BSEC-invalid state
0x23: BSEC-invalid state, with an active confirmed tamper triggered in the device
The BSEC state is evaluated each time INIT_DONE is set in BSEC_OTPSR.
For more information, refer to *Section 6.3.7: Lifecycle management*

Bits 25:2 Reserved, must be kept at reset value.

Bit 1 **HVALID**: Hardware key valid
0: Derived hardware unique key (DHUK) feature cannot be used in SAES peripheral.
1: Derived hardware unique key (DHUK) feature can be used in SAES peripheral.

Bit 0 Reserved, must be kept at reset value.

6.5.16 BSEC OTP status register (BSEC_OTPSR)

Address offset: 0xE44

Reset value: 0x0000 0000

This register is refreshed each time a fuse operation is triggered through BSEC_OTPCR.

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	PPLMF	PPLF	SECF	DEDF	DISTURBF	PROGFAIL
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPSEC	OTPERR	OTPNVIR	Res.	HIDEUP	INIT_DONE	BUSY
									r	r	r		r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **AMEF**: Addresses mismatch error flag

This bit is set when the address used by BSEC to read a fuse word is different from the programmed address (either by BSEC or by the application). Such event should never happen in normal BSEC operations.

Bit 21 **PPLMF**: Permanent programming lock mismatch flag

This bit is set when the two words constituting a lower fuse word are reporting different permanent programming lock statuses after BSEC reset or after the latest read.
For more details about permanent programming lock, refer to PPLOCK in BSEC_OTPCR.

Bit 20 **PPLF**: Permanent programming lock flag

This bit is set when the most recently read word is detected permanently locked.
For more details about permanent programming lock, refer to PPLOCK in BSEC_OTPCR.

Bit 19 **SECF**: Single error correction flag

This bit is set when a single-bit-error correction is detected after BSEC reset or after the latest read.

Bit 18 **DEDF**: Double error detection flag

This bit is set when a double error is detected after BSEC reset or after the latest read.
When DEDF bit is set after reading fuse word w, the value in BSEC_FVRw is the one read from OTP memory.

Bit 17 **DISTURBF**: Disturb flag

This bit is set when an unexpected error occurred after BSEC reset or after the latest read.
When DISTURBF bit is set after reading fuse word w, the value in BSEC_FVRw is the one read from OTP memory.

Bit 16 **PROGFAIL**: Programming failed

This bit is set when last OTP programming by BSEC failed.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **OTPSEC**: OTP with single error correction

This bit is set when at least one single-error correction is detected during BSEC reset operations. In this case shadow registers are usable.

Bit 5 **OTPERR**: OTP with error

This bit is set when at least one error is detected during BSEC reset operations.

Bit 4 **OTPNVIR**: OTP not virgin

This bit is set when first fuse word 0 is detected non-0 after BSEC reset operations.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **HIDEUP**: Hide upper fuse words

This bit is set when application secret fuse words 256 to 383 becomes physically inaccessible, that is functionally hidden fuse words read as 0 and programming requests are ignored.

Bit 1 **INIT_DONE**: Initialization done

This bit is set when BSEC reset operations are completed. All shadowed fuses are readable in BSEC_FVRw registers, and status bits in BSEC_SR and BSEC_OTPSR are up to date.

Bit 0 **BUSY**: Busy flag

0: BSEC is idle.

1: BSEC is busy.

6.5.17 BSEC epoch register (BSEC_EPOCHRx)

Address offset: 0xE80 + 0x4 * x (x = 0 to 1)

Reset value: 0x0000 0000

Writeable by boot ROM only, readable by secure privileged trusted domain or BootROM code only. Illegal reads return 0, illegal writes are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPOCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:0 **EPOCH[31:0]**: epoch

This value is wired out to the SAES peripheral, depending on the EPOCH_SEL bit in BSEC_EPOCHSELCR.

6.5.18 BSEC epoch selection control register (BSEC_EPOCHSELCR)

Address offset: 0xE88

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH_SEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EPOCH_SEL**: Selection of the epoch value to be sent to the SAES
0: SAES peripheral uses BSEC_EPOCHR0 as EPOCH value.
1: SAES peripheral uses BSEC_EPOCHR1 as EPOCH value.

6.5.19 BSEC debug control register for Cortex-M (BSEC_DBGMCR)

Address offset: 0xE8C

Reset value: 0x0000 0000

Secure privileged read and write by trusted domain only. Writeable only on a BSEC-closed device, once per warm reset. Illegal reads return 0, illegal writes are ignored. Reset to zero on *bsec_rst* (warm reset).

The AUTH_HDPL byte field indicates a temporal isolation level, as follows: the value 0xB4 means level 0, the value 0x51 means level 1, the value 0x8A means level 2, and the value 0x6F means level 3. All other byte values are undefined.

On a BSEC-closed device, debug is only possible on Cortex-M
if BSEC_DBGMCR[UNLOCK] = 0xB4, and the HDPL level indicated in BSEC_HDPLMSR is non-zero and reaches or surpasses that indicated by BSEC_DBGMCR[AUTH_HDPL]. In this case, secure debug is only possible if BSEC_DBGMCR[AUTH_SEC] = 0xB4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_SEC[7:0]								AUTH_HDPL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNLOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:24 **AUTH_SEC[7:0]:**
0xB4: Cortex-M secure debug possible when UNLOCK = 0xB4 and the level indicated in BSEC_HDPLMSR is greater than or equal to the level indicated by AUTH_HDPL.
Others: Cortex-M secure debug not possible (provided state is not BSEC-open)

Bits 23:16 **AUTH_HDPL[7:0]:** level at which debug may be opened.
0xB4: Level 0
0x51: Level 1
0x8A: Level 2
0x6F: Level 3
Others: undefined

Bits 15:8 **UNLOCK[7:0]:**
0xB4: Cortex-M nonsecure debug possible when the level indicated in BSEC_HDPLMSR is greater than or equal to the level given by AUTH_HDPL
Others: Cortex-M debug not possible (provided state is not BSEC-open)

Bits 7:0 Reserved, must be kept at reset value.

6.5.20 BSEC AP unlock (BSEC_AP_UNLOCK)

Address offset: 0xE90

Reset value: 0x0000 0000

Secure privileged read and write only. Writeable only on a BSEC-closed device, once per warm reset. Illegal reads return 0, illegal writes are ignored. Reset to zero on *bsec_srst* (cold reset).

When the state is BSEC-open, the *ap_unlocked* output to the DBG_MCU block is set to 0xB4. and the register is zeroed. When the BSEC state is not BSEC-open, it takes the value of the 8-bit UNLOCK field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **UNLOCK[7:0]:**
0xB4: Unlock the DBG_MCU AP interface.
Others: Do not unlock.

6.5.21 BSEC HDPL for Cortex-M status register (BSEC_HDPLMSR)

Address offset: 0xE94

Reset value: 0x0000 00B4

Secure privileged read only, by trusted domain or Cortex-M. Illegal reads return 0, writes are ignored. Reset to 0xB4 on *bsec_rst* (warm reset).

When coding a value of less than 3, it can be incremented by writing 0x60B166E7 to BSEC_HDPLMCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]:** current HDPL for Cortex-M
0xB4: Level 0
0x51: Level 1
0x8A: Level 2
0x6F: Level 3
Others: undefined

6.5.22 BSEC HDPL for Cortex-M control register (BSEC_HDPLMCR)

Address offset: 0xE98

Reset value: 0x0000 0000

Secure privileged write only by trusted domain or Cortex-M. Illegal reads return 0, writes are ignored.

When BSEC_HDPLMSR codes a value of less than 3, BSEC_HDPLMSR can be incremented by writing 0x60B166E7 to BSEC_HDPLMCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INCR_HDPL[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INCR_HDPL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **INCR_HDPL[31:0]**: Increment HDPL for Cortex-M
When HDPL is less than 3, writing 0x60B1 66E7 increments HDPL by one in BSEC_HDPLMSR.
Otherwise, writes are ignored. Reads return zero.

6.5.23 BSEC next HDPL control register for Cortex-M (BSEC_NEXTHDPLMCR)

Address offset: 0xE9C

Reset value: 0x0000 0000

Secure privileged read and write only, by trusted domain or Cortex-M. Illegal reads return 0, writes are ignored. Reset to 0 on *bsec_rst* (warm reset).

This value determines the amount to add to the level indicated by BSEC_HDPLMSR to output to the SAES on obk_hdpl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **INCR[1:0]**: Increment

Amount to add to the level indicated by BSEC_HDPLMSR to generate *obk_hdpl[7:0]* output (clamped at 3).

6.5.24 BSEC debug control register for Cortex-A (BSEC_DBGACR)

Address offset: 0xEAC

Reset value: 0x0000 0000

Secure privileged access by trusted domain only. Writeable only on a BSEC-closed device, once per warm reset. Illegal reads return 0, illegal writes are ignored. Reset to zero on *bsec_rst* (warm reset).

The AUTH_HDPL byte field indicates a temporal isolation level, as follows: the value 0xB4 means level 0, the value 0x51 means level 1, the value 0x8A means level 2, and the value 0x6F means level 3. All other byte values are undefined.

On a BSEC-closed device, debug is only possible on Cortex-A if BSEC_DBGACR[UNLOCK] = 0xB4, and the HDPL level indicated in BSEC_HDPLASR is non-zero and reaches or surpasses that indicated by BSEC_DBGACR[AUTH_HDPL]. In this case, secure debug is only possible if BSEC_DBGACR[AUTH_SEC] = 0xB4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_SEC[7:0]								AUTH_HDPL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNLOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:24 **AUTH_SEC[7:0]:**

0xB4: Cortex-A secure debug possible when UNLOCK = 0xB4 and the level indicated in BSEC_HDPLASR is greater than or equal to the level indicated by AUTH_HDPL

Others: secure debug not possible (provided state is not BSEC-open)

Bits 23:16 **AUTH_HDPL[7:0]:** level at which Cortex-A debug may be opened.

0xB4: Level 0

0x51: Level 1

0x8A: Level 2

0x6F: Level 3

Others: undefined

Bits 15:8 **UNLOCK[7:0]:**

0xB4: Cortex-A nonsecure debug possible when the level indicated in BSEC_HDPLASR is greater than or equal to the level given by AUTH_HDPL

Others: Cortex-A debug not possible (provided state is not BSEC-open)

Bits 7:0 Reserved, must be kept at reset value.

6.5.25 BSEC HDPL for Cortex-A status register (BSEC_HDPLASR)

Address offset: 0xEB4

Reset value: 0x0000 00B4

Secure privileged read only, by trusted domain or Cortex-A. Illegal reads return 0, writes are ignored. Reset to 0xB4 on *bsec_hrst* (hot reset).

When coding a value of less than 3, it can be incremented by writing 0x60B166E7 to BSEC_HDPLACR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]:** current HDPL for Cortex-A

0xB4: Level 0

0x51: Level 1

0x8A: Level 2

0x6F: Level 3

Others: undefined

6.5.26 BSEC HDPL for Cortex-A control register (BSEC_HDPLACR)

Address offset: 0xEB8

Reset value: 0x0000 0000

Secure privileged write only, by trusted domain or Cortex-A. Illegal reads return 0, writes are ignored.

When BSEC_HDPLASR codes a value of less than 3, BSEC_HDPLASR can be incremented by writing 0x60B166E7 to BSEC_HDPLACR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INCR_HDPL[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INCR_HDPL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **INCR_HDPL[31:0]**: Increment HDPL for Cortex-A

When HDPL is less than 3, writing 0x60B1 66E7 increments HDPL by 1 in BSEC_HDPLASR.

Otherwise, writes are ignored. Reads return zero.

6.5.27 BSEC next HDPL control register for Cortex-A (BSEC_NEXTHDPLACR)

Address offset: 0xEBC

Reset value: 0x0000 0000

Secure privileged read and write only, by trusted domain or Cortex-A. Illegal reads return 0, writes are ignored. Reset to zero on *bsec_hrst* (hot reset)

This value determines the amount to add to the level indicated by BSEC_HDPLASR to output to the SAES on obk_hdpl.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **INCR[1:0]**: Increment

Amount to add to the level indicated by BSEC_HDPLASR to generate *obk_hdpl[7:0]* output (clamped at 3).

6.5.28 BSEC write once scratch register x (BSEC_WOSCRx)

Address offset: 0xF40 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is reset when *bsec_rst* is asserted.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WOSDATA[31:16]															
rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOSDATA[15:0]															
rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo

Bits 31:0 **WOSDATA[31:0]**: Write once scratch data

ROM code can use this register to store values that needs to be persistent across hot resets. It is writable once, reset by either cold or warm BSEC reset.

6.5.29 BSEC hot reset count register (BSEC_HRCR)

Address offset: 0xFE8

Reset value: 0x0000 0000

This register is reset when *bsec_rst* is asserted.

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HRC[31:0]**: Hot reset counter

This bitfield counts the number of hot resets since the last BSEC cold/warm reset.
See *Section 6.3.3: BSEC reset and clocks* for details about resets.

6.5.30 BSEC warm reset count register (BSEC_WRCR)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is reset when *bsec_srst* is asserted.

Secure privileged read only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRC[31:0]**: Warm reset counter

This bitfield counts the number of warm resets since the last BSEC cold reset.
See *Section 6.3.3: BSEC reset and clocks* for details about resets.

6.5.31 BSEC version register (BSEC_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0012

Secure privileged read access only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

6.5.32 BSEC identification register (BSEC_IPIDR)

Address offset: 0xFF8

Reset value: 0x0010 0033

Secure privileged read access only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: BSEC identification

6.5.33 BSEC size identification register (BSEC_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

Secure privileged read access only (return 0 otherwise). Only trusted domain or Boot ROM code can access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: BSEC size identification

This bitfield returns the size identification code of the BSEC IP as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x04 (4 Kbytes address decoding)

6.5.34 BSEC register map

Table 32. BSEC register map and reset values

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4 * w, (w=0 to 375)	**BSEC_FVRw**	FV[31:0]																															
	Reset value(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x5E0 - 0x7FC	Reserved	Res.																															
0x800 + 0x4 * x, (x=0 to 11)	**BSEC_ SPLOCKx**	SPLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32. BSEC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x830 - 0x83C	Reserved	Res.																															
0x840 + 0x4 * x (x=0 to 11)	**BSEC_ SWLOCKx**	SWLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x870 - 0x87C	Reserved	Res.																															
0x880 + 0x4 * x (x=0 to 11)	**BSEC_ SRLOCKx**	SRLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8B0 - 0x8BC	Reserved	Res.																															
0x8C0 + 0x4 * x (x=0 to 11)	**BSEC_ OTPVLDRx**	VLDF{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8F0 - 0x93C	Reserved	Res.																															
0x940 + 0x4 * x (x=0 to 11)	**BSEC_SFSRx**	SFW{i + 32 * x} (i = 31 to 0)																															
	Reset value[1]	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	x	x	x	x	0	x	0	x	x	x	x	x	x	x	x	x	x
0x970 - 0xC00	Reserved	Res.																															
0xC04	**BSEC_OTPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LASTCID [2:0]			Res.	Res.	Res.	Res.	PPLOCK	PROG	Res.	Res.	Res.	Res.	ADDR[8:0]								
	Reset value											0	0	0					0	0					0	0	0	0	0	0	0	0	0
0xC08	**BSEC_WDR**	WRDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0C - 0xDFC	Reserved	Res.																															
0xE00 + 0x4 * x (x=0 to 3)	**BSEC_ SCRATCHRx**	SDATA[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xE10	**BSEC_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HKLOCK	DENLOCK	GWLOCK
	Reset value																														0	0	0
0xE14	**BSEC_ JTAGINR**	JDATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE18	**BSEC_ JTAGOUTR**	JDATAOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE1C	Reserved	Reserved																															
0xE20	BSEC_DENR	PATTERN												Res.	Res.	RCODE5	RCODE4	RCODE3	RCODE2	RCODE1	RCODE0	SPNIDENM	SPIDENM	NIDENM	DBGENM	DBGSWEN	SPNIDENA	SPIDENA	HDPEN	DEVICEEN	NIDENA	DBGENA	Res.
	Reset value[1]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xE24	**BSEC_ UNMAPR**	UNMAP[31:0]																															
	Reset value[1]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xE28 - 0xE3C	Reserved	Res.																															
0xE40	**BSEC_SR**	NVSTATE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HVALID	Res.
	Reset value	0	0	0	0	0	0																									0	

Table 32. BSEC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE44	**BSEC_OTPSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	PPLMF	PPLF	SECF	DEDF	DISTURBF	PROGFAIL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPSEC	OTPERR	OTPNVIR	Res.	HIDEUP	INIT_DONE	BUSY
	Reset value										0	0	0	0	0	0	0										0	0	0		0	0	0
0xE48 - 0xE7C	Reserved	Res.																															
0xE80 + 0x4 * x (x= 0 to 1)	**BSEC_ EPOCHRx**	EPOCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE88	**BSEC_ EPOCHSELCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH_SEL
	Reset value																																0
0xE8C	BSEC_ DBGMCR	AUTH_SEC[7:0]								AUTH_HDPL[7:0]								UNLOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0xE90	BSEC_ AP_UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xE94	BSEC_ HDPLMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									0	0	0	0	0	0	B	4
0xE98	BSEC_ HDPLMCR	INCR_HDPL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE9C	BSEC_NEXTHD PLMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR [1:0]	
	Reset value																															0	0
0xEA0 - 0xEA8	Reserved	Reserved																															
0xEAC	BSEC_ DBGACR	AUTH_SEC[7:0]								AUTH_HDPL[7:0]								UNLOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0xEB0	Reserved	Res.																															
0xEB4	BSEC_ HDPLASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									0	0	0	0	0	0	B	4
0xEB8	BSEC_ HDPLACR	INCR_HDPL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xEBC	BSEC_ NEXTHDPLAC R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR [1:0]	
	Reset value																															0	0
0xEC0 - 0xF3C	Reserved	Reserved																															
0xF40 + 0x4*x (x=0 to 7)	BSEC_ WOSCRx	WOSDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF60 - 0xFE4	Reserved	Res.																															
0xFE8	BSEC_HRCR	HRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFEC	BSEC_WRCR	WRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32. BSEC register map and reset values (continued)

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	**BSEC_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	1	0
0xFF8	**BSEC_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
0xFFC	**BSEC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

1. See register description.

Refer to *Section 2.4* for the register boundary addresses.

7 OTP mapping (OTP)

The STM32MP21xx devices have 12288 OTP (one-time programmable) bits which can be read-accessed in 384 words: BSEC_OTP_DATAx (x = 0 to 383).

The OTP bits are read or programmed by the BSEC (boot, security and OTP control). See *Section 6: Boot and security control (BSEC)*.

Before being programmed, the OTP bit is at zero. The OTP words are described in the table below.

OTP are organized into the following regions:

- lower OTP (OTP0 to OTP127)
- mid OTP (OTP128 to OTP255)
- upper OTP (OTP256 to OTP383)

The OTP word most relevant properties are listed in the following columns:

- Description: OTP word acronym and function
- Shadowed: OTP word is shadowed into a register after reset. A shadowed word may be read/write, read only, or no access, according to shadowed type column.
- Wired: OTP word is shadowed with a wired connection in the device.
- D1STDBY: OTP is used by the boot ROM at D1 Standby exit.
- Owner: ST side: OTP word is programmed by STMicroelectronics at factory level.
  Customer side: OTP can be programmed by the application.
- Shadowed type: A shadow register can be read/write, read only, no access.

Table 33. STM32MP21xx OTP mapping

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
Lower OTP region						
OTP0	OTP_HW_WORD0	Yes	Yes	Yes (BSEC)	ST/ Customer	Read only
OTP1	OTP_HW_WORD1	Yes	N/A	Yes (BSEC)	ST	Read only
OTP2	OTP_HW_WORD2	Yes	N/A	Yes (BSEC)	ST	Read only
OTP3	OTP_HW_WORD3	Yes	N/A	Yes (BSEC)	ST	Read only
OTP4	OTP_HW_WORD4	Yes	Yes	Yes (BSEC)	ST	Read only
OTP5	ID0	Yes	No	No	ST	Read only
OTP6	ID1	Yes	No	No	ST	Read only
OTP7	ID2	Yes	No	No	ST	Read only
OTP8	OTP_CID_PROV	Yes	No	No	ST	Read only
OTP9	RPN_CODING	Yes	Yes	Yes (SoC)	ST	Read only
OTP10	BOOTROM_CONFIG_1	No	Yes	No	ST	-
OTP11	BOOTROM_CONFIG_2	Yes	Yes	No	Customer	Read/write
OTP12	BOOTROM_CONFIG_3	No	Yes	No	Customer	-

Table 33. STM32MP21xx OTP mapping (continued)

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
OTP13	Reserved by ST	Yes	No	No	ST	Read only
OTP14	Reserved by ST	Yes	No	No	ST	Read only
OTP15	Reserved by ST	Yes	No	No	ST	Read only
OTP16	BOOTROM_CONFIG_7	Yes	Yes	No	Customer	Read/write
OTP17	BOOTROM_CONFIG_8	No	Yes	No	Customer	-
OTP18	BOOTROM_CONFIG_9	No	Yes	No	Customer	-
OTP19	BOOTROM_CONFIG_10	No	Yes	No	Customer	-
OTP20	BOOTROM_CONFIG_11	No	Yes	No	Customer	-
OTP21	BOOTROM_CONFIG_12	No	Yes	No	Customer	-
OTP22	BOOTROM_CONFIG_13	No	Yes	No	Customer	-
OTP23	H32ENCPRVKEY	No	Yes	No	ST	-
OTP24	BOOTROM_TZ_EPOCH0	No	No	No	Customer	-
OTP25	BOOTROM_TZ_EPOCH1	No	No	No	Customer	-
OTP26	BOOTROM_TZ_EPOCH2	No	No	No	Customer	-
OTP27	BOOTROM_TZ_EPOCH3	No	No	No	Customer	-
OTP28	BOOTROM_TZ_EPOCH4	No	No	No	Customer	-
OTP29	BOOTROM_TZ_EPOCH5	No	No	No	Customer	-
OTP30	BOOTROM_TZ_EPOCH6	No	No	No	Customer	-
OTP31	BOOTROM_TZ_EPOCH7	No	No	No	Customer	-
OTP32	BOOTROM_NS_EPOCH0	No	No	No	Customer	-
OTP33	BOOTROM_NS_EPOCH1	No	No	No	Customer	-
OTP34	BOOTROM_NS_EPOCH2	No	No	No	Customer	-
OTP35	BOOTROM_NS_EPOCH3	No	No	No	Customer	-
OTP36	BOOTROM_NS_EPOCH4	No	No	No	Customer	-
OTP37	BOOTROM_NS_EPOCH5	No	No	No	Customer	-
OTP38	BOOTROM_NS_EPOCH6	No	No	No	Customer	-
OTP39	BOOTROM_NS_EPOCH7	No	No	No	Customer	-
OTP40 to OTP101	Available for customer	No	No	No	Customer	-
OTP102	ID	No	No	No	ST	-
OTP103	CRC_HSM	No	No	No	ST	-
OTP104	CAL1	No	No	No	ST	-
OTP105	Reserved	No	No	No	ST	-
OTP106	CAL3	No	No	No	ST	-

Table 33. STM32MP21xx OTP mapping (continued)

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
OTP107	Reserved	No	No	No	ST	-
OTP108	CAL5	No	No	No	ST	-
OTP109	Reserved	No	No	No	ST	-
OTP110	CAL7	No	No	No	ST	-
OTP111	Reserved	No	No	No	ST	-
OTP112	ENGI1	No	No	No	ST	-
OTP113	ENGI2	No	No	No	ST	-
OTP114	ENGI3	No	No	No	ST	-
OTP115	ENGI4	No	No	No	ST	-
OTP116	ENGI5	No	No	No	ST	-
OTP117	ENGI6	No	No	No	ST	-
OTP118	ENGI7	No	No	No	ST	-
OTP119	ENGI8	No	No	No	ST	-
OTP120	ATRIM1	Yes	No	hws[31:0]	ST	Read/write
OTP121	ATRIM2	Yes	No	hws[63:32]	ST	Read/write
OTP122	ATRIM3	Yes	No	hws[95:64]	ST	Read/write
OTP123	ATRIM4	Yes	No	hws[127:96]	ST	Read/write
OTP124	HCONF1	Yes	No	hws[159:128]	Customer	Read/write
OTP125	MREPAIR1	Yes	No	hws[191:160]	ST	Read/write
OTP126	MREPAIR2	Yes	No	hws[223:192]	ST	Read/write
OTP127	MREPAIR3	Yes	No	hws[255:224]	ST	Read/write
-						
Mid OTP region						
OTP128	STM32CERTIF0	No	No	No	ST	-
OTP129	STM32CERTIF1	No	No	No	ST	-
OTP130	STM32CERTIF2	No	No	No	ST	-
OTP131	STM32CERTIF3	No	No	No	ST	-
OTP132	STM32CERTIF4	No	No	No	ST	-
OTP133	STM32CERTIF5	No	No	No	ST	-
OTP134	STM32CERTIF6	No	No	No	ST	-
OTP135	STM32CERTIF7	No	No	No	ST	-
OTP136	STM32CERTIF8	No	No	No	ST	-
OTP137	STM32CERTIF9	No	No	No	ST	-
OTP138	STM32CERTIF10	No	No	No	ST	-

Table 33. STM32MP21xx OTP mapping (continued)

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
OTP139	STM32CERTIF11	No	No	No	ST	-
OTP140	STM32CERTIF12	No	No	No	ST	-
OTP141	STM32CERTIF13	No	No	No	ST	-
OTP142	STM32CERTIF14	No	No	No	ST	-
OTP143	STM32CERTIF15	No	No	No	ST	-
OTP144	STM32CERTIF16	No	No	No	ST	-
OTP145	STM32CERTIF17	No	No	No	ST	-
OTP146	STM32CERTIF18	No	No	No	ST	-
OTP147	STM32CERTIF19	No	No	No	ST	-
OTP148	STM32CERTIF20	No	No	No	ST	-
OTP149	STM32CERTIF21	No	No	No	ST	-
OTP150	STM32CERTIF22	No	No	No	ST	-
OTP151	STM32CERTIF23	No	No	No	ST	-
OTP152	OEM_KEY1_ROT0	No	No	No	Customer	-
OTP153	OEM_KEY1_ROT1	No	No	No	Customer	-
OTP154	OEM_KEY1_ROT2	No	No	No	Customer	-
OTP155	OEM_KEY1_ROT3	No	No	No	Customer	-
OTP156	OEM_KEY1_ROT4	No	No	No	Customer	-
OTP157	OEM_KEY1_ROT5	No	No	No	Customer	-
OTP158	OEM_KEY1_ROT6	No	No	No	Customer	-
OTP159	OEM_KEY1_ROT7	No	No	No	Customer	-
OTP160	OEM_KEY2_ROT0	No	No	No	ST	-
OTP161	OEM_KEY2_ROT1	No	No	No	ST	-
OTP162	OEM_KEY2_ROT2	No	No	No	ST	-
OTP163	OEM_KEY2_ROT3	No	No	No	ST	-
OTP164	OEM_KEY2_ROT4	No	No	No	ST	-
OTP165	OEM_KEY2_ROT5	No	No	No	ST	-
OTP166	OEM_KEY2_ROT6	No	No	No	ST	-
OTP167	OEM_KEY2_ROT7	No	No	No	ST	-
OTP168	STM32PUBKEY0	No	No	No	ST	-
OTP169	STM32PUBKEY1	No	No	No	ST	-
OTP170	STM32PUBKEY2	No	No	No	ST	-
OTP171	STM32PUBKEY3	No	No	No	ST	-
OTP172	STM32PUBKEY4	No	No	No	ST	-

Table 33. STM32MP21xx OTP mapping (continued)

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
OTP173	STM32PUBKEY5	No	No	No	ST	-
OTP174	STM32PUBKEY6	No	No	No	ST	-
OTP175	STM32PUBKEY7	No	No	No	ST	-
OTP176	STM32PUBKEY8	No	No	No	ST	-
OTP177	STM32PUBKEY9	No	No	No	ST	-
OTP178	STM32PUBKEY10	No	No	No	ST	-
OTP179	STM32PUBKEY11	No	No	No	ST	-
OTP180 to OTP254	Available to customer	No	No	No	Customer	-
OTP255	ST_RSSE_EDMK_DERIV_CSTE_FUSE	No	No	No	ST	-
Upper OTP region						
OTP256	OTP_RMA_LOCK_PSWD0	Yes	No	No	Customer	-
OTP257	OTP_RMA_LOCK_PSWD1	Yes	No	No	Customer	-
OTP258	OTP_RMA_LOCK_PSWD2	Yes	No	No	Customer	-
OTP259	OTP_RMA_LOCK_PSWD3	Yes	No	No	Customer	-
OTP260 to OTP347	OEM Secrets available to customer	No	No	No	Customer	-
OTP348	OEM_KEY2_EDMK0	No	No	No	Customer	-
OTP349	OEM_KEY2_EDMK1	No	No	No	Customer	-
OTP350	OEM_KEY2_EDMK2	No	No	No	Customer	-
OTP351	OEM_KEY2_EDMK3	No	No	No	Customer	-
OTP352	OEM_KEY2_EDMK4	No	No	No	Customer	-
OTP353	OEM_KEY2_EDMK5	No	No	No	Customer	-
OTP354	OEM_KEY2_EDMK6	No	No	No	Customer	-
OTP355	OEM_KEY2_EDMK7	No	No	No	Customer	-
OTP356	OEM_KEY1_EDMK0	No	Yes	No	Customer	-
OTP357	OEM_KEY1_EDMK1	No	Yes	No	Customer	-
OTP358	OEM_KEY1_EDMK2	No	Yes	No	Customer	-
OTP359	OEM_KEY1_EDMK3	No	Yes	No	Customer	-
OTP360	OEM_KEY1_EDMK4	No	Yes	No	Customer	-
OTP361	OEM_KEY1_EDMK5	No	Yes	No	Customer	-
OTP362	OEM_KEY1_EDMK6	No	Yes	No	Customer	-
OTP363	OEM_KEY1_EDMK7	No	Yes	No	Customer	-
OTP364	STM32ENCPRVKEY0	No	No	No	ST	-

Table 33. STM32MP21xx OTP mapping (continued)

Word	Description	Shadowed	D1STDBY exit	Wired	Owner	Shadowed type
OTP365	STM32ENCPRVKEY1	No	No	No	ST	-
OTP366	STM32ENCPRVKEY2	No	No	No	ST	-
OTP367	STM32ENCPRVKEY3	No	No	No	ST	-
OTP368	STM32ENCPRVKEY4	No	No	No	ST	-
OTP369	STM32ENCPRVKEY5	No	No	No	ST	-
OTP370	STM32ENCPRVKEY6	No	No	No	ST	-
OTP371	STM32ENCPRVKEY7	No	No	No	ST	-
OTP372	STM32ENCPRVKEY8	No	No	No	ST	-
OTP373	STM32ENCPRVKEY9	No	No	No	ST	-
OTP374	STM32ENCPRVKEY10	No	No	No	ST	-
OTP375	STM32ENCPRVKEY11	No	No	No	ST	-
OTP376	HWKEY0	Yes	N/A	Yes (SAES)	ST	No access
OTP377	HWKEY1	Yes	N/A	Yes (SAES)	ST	No access
OTP378	HWKEY2	Yes	N/A	Yes (SAES)	ST	No access
OTP379	HWKEY3	Yes	N/A	Yes (SAES)	ST	No access
OTP380	HWKEY4	Yes	N/A	Yes (SAES)	ST	No access
OTP381	HWKEY5	Yes	N/A	Yes (SAES)	ST	No access
OTP382	HWKEY6	Yes	N/A	Yes (SAES)	ST	No access
OTP383	HWKEY7	Yes	N/A	Yes (SAES)	ST	No access

The following OTP words are described in details in *Table 34: STM32MP21xx OTP fuse description (lower OTP region)* or by BSEC:

- OTP0 to OTP4 are used by BSEC for the security life cycle (see *Section 6: Boot and security control (BSEC)*).
- OTP9 is used to code device part number (see *Section 81: Device electronic signature*).
- OTP10 to OTP22 are used for the boot ROM configuration.
- OTP104 to OTP 111 are used for analog calibration parameters set in ST factory.
- OTP112 to OTP 119 are used for engineering purposes and are set in ST factory.
- OTP120 to OTP123 are used for settings such as analog trimming and are set in ST factory.
- OTP124 is used for the product configuration by the user.
- OTP125 to OTP 127 are used for memory repair and are set in ST factory.

Table 34: STM32MP21xx OTP fuse description (lower OTP region) provides the bit-level description and other word properties in the following columns:

- wplock:
  - y: permanent programming lock by factory programming

Key storage in OTP

Keys are represented as a string of byte to be stored in consecutive OTP words. For example a 64-bit key (0xAABBCCDDEEFF5566) is stored into two consecutive OTP words KEY0 and KEY1.

A key is stored in OTP words using one of the following formats

- Format 1 KEY0 = 0xAABBCCDD, KEY1 = 0xEEFF5566
- Format 2 KEY0 = 0xDDCCBBAA, KEY1 = 0x6655FFEE
- Format 3 KEY0 = 0xEEFF5566, KEY1 = 0xAABBCCDD

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1)

OTP words/ bits	Name/description	wplock(2)	Detailed description
OTP0	OTP_HW_WORD0	y	OTP check word (virgin -> non virgin)
OTP1	OTP_HW_WORD1	-	OTP security word to close security state BSECXW
OTP2	OTP_HW_WORD2	-	OTP word for re-opening (CLOSE->OPEN) via RMA password: RMA bits
OTP3	OTP_HW_WORD3	-	OTP Word for re-opening (CLOSE->OPEN) via RMA password: RMA tries bits
OTP4	OTP_HW_WORD4	-	OTP word for TK retries (ECIES)
OTP5	ID0	y	ID0 for engineering purposes
OTP6	ID1	y	ID1 for engineering purposes
OTP7	ID2	y	ID2 for engineering purposes
OTP8	OTP_CID_PROV	y	OTP word for CID provisioning control
7 to 0	prov_cid_switch_enabling	-	Enable/disable provisioning cid for rsse_fw 0x0F (enabled): provisioning cid switching is enabled for rsse_fw. 0xFF (disabled): provisioning cid switching is permanently disabled. OTP8[7:0] to be programmed to 0x0F for life cycle state CLOSED_LOCKED_UNPROVD and OTP8[7:0] to be programmed to 0xFF for life cycle state CLOSED_LOCKED_PROVD.
31 to 8	Reserved	-	-
OTP9	RPN_CODING	y	-
OTP10	BOOTROM_CONFIG_1	-	-
0	chip_cpbk_y_sign	-	0 (positive): Device public key Y is positive 1 (negative): Device public key Y is negative
1	stkeyprov_ecies_ok	-	Status of ECIES ST key provisioning when it was attempted. 0 (failed): ECIES ST key provisioning last attempt was failed 1 (successful): ECIES ST key provisioning last attempt was successful

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
2	stkeyprov_hwkey_done	-	0 (no): HWKEY not provisioned 1 (yes): HWKEY was provisioned
3 to 6	security_counter	-	Security counter involved in productID for device certificate verification by HSM-OEM in RSSe_prov context
7 to 14	st_pub_key_id	-	ST ECDSA public key ID (ST key instance fuse part) involved in productID for device certificate verification by HSM-OEM in RSSe_prov context
15 to 18	rssefw_active_signing_key	-	[1-0xFF] → [1-8]: Value of monotonic counter is X where X is position of the most significant bit at 1. 8 possible ST public keys (ST key revocation feature for RSSe_FW authentication)
19 to 26	nb_added_stsecrets	-	0: no additional ST secrets [1-63]: number of OTP words located in upper area [360-nb_added_stsecrets..359] that were provisioned with ST secrets.
27 to 31	Reserved	-	-
OTP11	BOOTROM_CONFIG_2	-	boot source configuration word
0 to 4	Reserved	-	-
5	no_cpu_pll	-	0 (enabled): PLLs for CPU/AXI are enable for cold boot. 1 (disabled): PLLs for CPU/AXI are not enable for cold boot
6 to 9	Reserved	-	-
10 to 14	no_data_cache	-	0b00001: disables USART2 0b00010: disables UART5 0b00100: disables UART6
15	no_data_cache	-	0 (enabled): Data cache is used by boot ROM. 1 (disabled): Data cache is not used by boot ROM.
16 to 23	boot_source_disable	-	Each bit disables a boot source. Default to UART if all disabled. 0b00000001 : disable FMC boot source. 0b00000010 : disable QSPI NOR boot source. 0b00000100 : disable eMMC boot source. 0b00001000 : disable SD boot source. 0b00010000 : disable UART boot source. 0b00100000 : disable USB boot source. 0b01000000: disable QSPI NAND boot source. 0b10000000: disable QSPI HyperFlash boot source.
24 to 25	bootpins_layout_sel	-	[0-3]: Select one among the three possible BOOT pin layouts
26 to 29	boot_source_sel	-	[0-15]: Select one among the 16 possible boot source of the selected BOOT pin layout
30	dont_boot_on_cfm_tamper	-	0: boot on confirmed tamper 1: do not boot on confirmed tamper
31	Reserved	-	-

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
OTP12	BOOTROM_CONFIG_3	-	-
0 to 31	oem_fsbla_monotonic_counter	-	[1-0xFFFF] -> [1-32]: Value of monotonic counter is X where X is position of the most significant bit at 1.
OTP13	Reserved by ST	y	-
OTP14	Reserved by ST	y	-
OTP15	Reserved by ST	y	-
OTP16	BOOTROM_CONFIG_7	-	-
0	disable_traces	-	0 (no): boot ROM traces are enabled 1 (yes): boot ROM traces are disabled
1	disable_hse_freq_detect	-	0 (no): HSE frequency autodetection is enabled 1 (yes): HSE frequency autodetection is disabled
2	disable_hse_bypass_detect	-	0 (no): HSE bypass detection is enabled 1 (yes): HSE bypass detection is disabled
3	disable_blocking_failure_ traces	-	0 (no): blocking failure traces are enabled 1 (yes): blocking failure traces are disabled
4	a35_mode	-	0 (aarch64): FSBL-A is AArch64 1 (aarch32): FSBL-A is AArch32
5	fmc_force_sw_reset	-	FMC is used by CA35 to connect a NAND and by CM33 to connect a NOR or PSRAM 0 (no): Use RCC to reset FMC 1 (yes): Use software procedure to reset FMC without impacting CM33
6	emergency_debug_req	-	0 (no): emergency debug is not requested 1 (yes): emergency debug is requested
7	emmc_128k_boot_partition	-	0 (no): Boot ROM does not support *e*•MMC with 128Kb boot partition. 1 (yes): boot ROM supports *e*•MMC with 128 Kb boot partition.
8	fsbl_decrypt_prio	-	0 (speed): use CRYP (fast but no DPA protection) 1 (security): use SAES (slow but DPA protection)
9	Reserved	-	-
10	Reserved	-	-
11 to 13	HSE_value	-	0b000 (auto): HSE value is autodetected among 16, 20, 24, 28, 32, 36, 40, 48 MHz 0b001 (24 MHz): HSE = 24 MHz 0b010 (25 MHz): HSE = 25 MHz 0b011 (26 MHz): HSE = 26 MHz 0b100 (19_2 MHz): HSE = 19.2 MHz 0b101 (40 MHz): HSE = 40 MHz 0b110 (48 MHz): HSE = 48 MHz 0b111 (unused): Reserved
14	snand_need_plane_select_1	-	0 (no): Serial NAND plane select is not needed 1 (yes): Serial NAND plane select is need

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
15 to 17	snand_need_plane_select_1	-	0 (unset): ECC unset 1 (hamming): ECC 1bit (Hamming) 2 (bch4): ECC 4bit (BCH4) 3 (bch8): ECC 8bit (BCH8) 4 (ondie): on-die ECC
18	pnand_bus_width_1	-	0 (8-bits): data width is 8-bits 1 (16-bits): data width is 16-bits
19 to 26	pnand_bus_width_1	-	[1-256]: Number of block = 256 * value
27 to 28	nand_block_size_1	-	[1-4]: Block size in number of pages
29 to 30	nand_page_size_1	-	0 (64): 64 pages per block 1 (128): 128 pages per block 2 (256): 256 pages per block
31	pnand_param_stored_in_otp	-	0 (no): Boot ROM uses ONFI parameter table to get parallel nand parameters. 1 (yes): parallel nand parameters are defined in bank1 or bank2, depending on nand_config_distribution value.
OTP17	BOOTROM_CONFIG_8	-	-
0 to 7	oem_active_signing_key1	-	[1-256] -> [1-8]: Value of monotonic counter is X where X is position of the most significant bit at 1. 8 possible OEM public keys (OEM key revocation feature for OEM-FSBL authentication).
8	oem_keys2_enable	-	0 (no): keys2 are not used; keys1 are used for both FSBLA and FSBLM. 1 (yes): keys2 are enabled; keys1 are used for FSBLA; keys2 are used for FSBLM.
16 to 31	Reserved	-	-
OTP18	BOOTROM_CONFIG_9	-	-
0 to 3	secure_boot	-	0 (unlocked): device is in CLOSED_UNLOCKED state. Secure boot is not enforced (FSBL authentication is not mandatory). [1-15] (locked): device is in CLOSED_LOCKED state. Secure boot is enforced (FSBL authentication is mandatory).
4 to 7	prov_done	-	0: RSSE provisioning is not done [1-15]: RSSE provisioning is done
8 to 11	debug_lock	-	Lock debug enabling until next reset when device is CLOSED-LOCKED. 0: does not lock debug enabling [1-15]: locks debug enabling
12 to 15	otp_prov_done	-	0: RSSe OTP provisioning is not done [1-15]: RSSe OTP provisioning part done, used to manage RSSe extension issue
16 to 21	Reserved	-	-

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
22	ns_epoch_enable		0: Boot ROM does not manage NS_EPOCH. The BOOTROM_NS_EPOCH monotonic counter fuse bits are free for other use. 1: Boot ROM manages NS_EPOCH. The Boot ROM uses BOOTROM_NS_EPOCH monotonic counter fuse bits.
23 to 26	fingerprint_enable		0 (no): fingerprint feature is disabled 1 (yes): fingerprint feature is enabled
27 to 31	Reserved	-	-
OTP19	BOOTROM_CONFIG_10	-	-
0 to 31	oem_keys2_enable	-	Monotonic version number of OEM-FSBL [1-0xFFFF] -> [1-32]: Value of monotonic counter is X where X is position of the most significant bit at 1.
OTP20	BOOTROM_CONFIG_11	-	-
0	nand_config_distribution	-	0 (bk1_snand_bk2_pnand): pNAND config in nand*_2 fields / sNAND config in nand*_1 fields 1 (bk1_pnand_bk2_snand): pNAND config in nand*_1 fields / sNAND config in nand*_2 fields
1	snand_need_plane_select_2	-	idem BOOTROM_CONFIG_7.snand_need_plane_select_1
2 to 4	pnand_number_of_ecc_bits_2	-	idem BOOTROM_CONFIG_7.pnand_number_of_ecc_bits_1
5	pnand_bus_width_2	-	idem BOOTROM_CONFIG_7.pnand_bus_width_1
6 to 13	nand_nb_of_blocks_2	-	idem BOOTROM_CONFIG_7.nand_nb_of_blocks_1
14 to 15	nand_block_size_2	-	idem BOOTROM_CONFIG_7.nand_block_size_1
16 to 17	nand_page_size_2	-	idem BOOTROM_CONFIG_7.nand_page_size_1
18	hyperflash_3V3_device	-	is HyperFlash a 3.3 V device 0:no 1:yes
19 to 21	rng_htcr_value	-	6: 0xAAC7 value set by engineering 7: 0xA2B3
22	ospi_io_speed_ovrw	-	0 (no): OSPI io speed is not overwritten by otp configuration 1 (yes): OSPI io speed is overwritten by otp configuration
23 to 24	ospi_io_speed_clk_nclk	-	0b00 (low): low speed. 0b01 (medium): medium speed. 0b10 (high): high speed 0b11 (very_high): very high speed
25 to 26	ospi_io_speed_data_cs		0b00 (low): low speed. 0b01 (medium): medium speed. 0b10 (high): high speed 0b11 (very_high): very high speed
27 to 31	Reserved	-	-
OTP21	BOOTROM_CONFIG_12	-	-

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
0 to 31	rssefw_version_monotonic_ counter	-	Monotonic version number of RSSe FW [1-0xFFFF] -> [1-32]: Value of monotonic counter is X where X is position of the most significant bit at 1.
OTP22	BOOTROM_CONFIG_13	-	-
0 to 7	oem_active_signing_key2	-	[1-256] -> [1-8]: Value of monotonic counter is X where X is position of the most significant bit at 1. 8 possible OEM public keys (OEM key revocation feature for OEM-FSBL authentication).
8 to 31	Reserved	-	-
OTP23	H32ENCPRVKEY	-	Last 32 bits of SHA256(STM32ENCPRVKEY)
OTP24	BOOTROM_TZ_EPOCH0	-	Secure side Epoch counter, part1
OTP25	BOOTROM_TZ_EPOCH1	-	Secure side Epoch counter, part2
OTP26	BOOTROM_TZ_EPOCH2	-	Secure side Epoch counter, part3
OTP27	BOOTROM_TZ_EPOCH3	-	Secure side Epoch counter, part4
OTP28	BOOTROM_TZ_EPOCH4	-	Secure side Epoch counter, part5
OTP29	BOOTROM_TZ_EPOCH5	-	Secure side Epoch counter, part6
OTP30	BOOTROM_TZ_EPOCH6	-	Secure side Epoch counter, part7
OTP31	BOOTROM_TZ_EPOCH7	-	Secure side Epoch counter, part8
OTP32	BOOTROM_NS_EPOCH0	-	Non Secure Epoch counter, part1
OTP33	BOOTROM_NS_EPOCH1	-	Non Secure Epoch counter, part2
OTP34	BOOTROM_NS_EPOCH2	-	Non Secure Epoch counter, part3
OTP35	BOOTROM_NS_EPOCH3	-	Non Secure Epoch counter, part4
OTP36	BOOTROM_NS_EPOCH4	-	Non Secure Epoch counter, part5
OTP37	BOOTROM_NS_EPOCH5	-	Non Secure Epoch counter, part6
OTP38	BOOTROM_NS_EPOCH6	-	Non Secure Epoch counter, part7
OTP39	BOOTROM_NS_EPOCH7	-	Non Secure Epoch counter, part8
OTP40 to OTP101	Reserved	-	-
OTP102	ID	-	-
0 to 5	REV_ID	-	Revision identification
6 to 31	Reserved ST	-	-
OTP103	CRC_HSM	y	ST Reserved
OTP104	CAL1	y	-
0 to 15	A coefficient for temperature sensor 1	-	No calibration: +/-3C Mode 1 calibration: ±1 °C (0...70) and ±1.25 °C (-40...125)

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
16 to 31	B coefficient for temperature sensor 1	-	-
OTP105	Reserved for ST	-	-
OTP106	CAL3	y	-
0 to 4	CPU_SVT16	-	CPU_SVT16 process code
5 to 9	CPU_LVT16	-	CPU_LVT16 process code
10 to 14	CPU_ULVT16	-	CPU_ULVT16 process code
15	Reserved for ST	-	-
16 to 20	CPU_SVT20	-	CPU_SVT20 process code
21 to 25	CPU_LVT20	-	CPU_LVT20 process code
26 to 30	CPU_ULVT20	-	CPU_ULVT20 process code
31	Reserved for ST	-	-
OTP107	Reserved for ST	-	-
OTP108	CAL5	-	-
OTP109	Reserved for ST	y	-
OTP110	CAL7	y	-
0 to 15	VREFIN_CAL	-	Raw data acquired by ADC1 at temperature of 30 °C
16 to 31	Reserved for ST	-	Reserved for ADC Calibration
OTP111	Reserved for ST	-	-
OTP112	ENGI1	y	Reserved for ST
OTP113	ENGI2	y	Reserved for ST
OTP114	ENGI3	y	Reserved for ST
OTP115	ENGI4	y	Reserved for ST
OTP116	ENGI5	y	Reserved for ST
OTP117	ENGI6	y	Reserved for ST
OTP118	ENGI7	y	Reserved for ST
OTP119	ENGI8	y	Reserved for ST
OTP120	ATRIM1	y	-
0 to 5	BG_ TRIM[5:0]	-	Internal bandgap trim
6 to 11	VREF_ TRIM_1V5[5:0]	-	ADC reference trim for 1.5 V
12 to 17	LSI_TRIM[5:0]	-	LSI frequency trim
18 to 22	TMON_TRIM[4:0]	-	Temperature monitor trim
23 to 24	BKPREG_TRIM[1:0]	-	Backup regulator (.8 V) adjust

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
25 to 30	VREF_ TRIM_1V2[5:0]	-	ADC reference trim for 1.21 V
31	Reserved for ST	-	-
OTP121	ATRIM2	Y	-
0 to 7	MSI_16M_ TRIM[7:0]	-	MSI frequency trim at 16 MHz
8 to 15	Reserved for ST	-	-
16 to 24	HSI_ TRIM[8:0]	-	HSI frequency trim
25 to 31	Reserved for ST	-	-
OTP122	ATRIM3	Y	-
0 to 2	PKG[2:0]	-	Package options: 000: Custom or Die503 001: VFBGA361 (10 × 10 pitch 0.5 mm) 011: VFBGA273 (11 × 11 pitch 0.5 mm) 100: VFBGA225 (8 × 8 pitch 0.5 mm) 101: VFBGA264 (14 × 14 pitch 0.8 mm)
3 to 19	Reserved for ST	-	-
20 to 28	BSEC_ TRIM[8:0]	-	BSEC local oscillator frequency trim at EWS
29 to 30	Reserved for ST	-	
31	USE_AMBIANT	-	Use BSEC_AMBIANT_TRIM for BSEC local oscillator
OTP123	ATRIM4	Y	-
0 to 3	Reserved for ST	-	-
4 to 7	OCTOSPI1_IOCOMP	-	OCTOSPI1 IO segment compensation trim
8 to 11	EMMC_IOCOMP	-	EMMC IO segment compensation trim
12 to 15	SD_IOCOMP	-	SD IO segment compensation trim
16 to 19	VDD_IOCOMP	-	V_{DD} IO segment compensation trim
20 to 22	HSI_TEMPSEL[2:0]	-	Temperature correction adjustment during validation
23 to 31	BSEC_AMBIANT_TRIM[8:0]	-	BSEC local oscillator frequency trim at ambient temperature
OTP124	HCONF1	-	-
0	IWDG1_HW	-	IWDG1 start on reset
1	IWDG1_FZ_STOP	-	IWDG1 freeze in Stop mode
2	IWDG1_FZ_STANDBY	-	IWDG1 freeze in Standby mode
3	IWDG2_HW	-	IWDG2 start on reset
4	IWDG2_FZ_STOP	-	IWDG2 freze in Stop mode
5	IWDG2_FZ_STANDBY	-	IWDG2 freeze in Standby mode
6	IWDG3_HW	-	IWDG3 start on reset
7	IWDG3_FZ_STOP	-	IWDG3 freeze in Stop mode

Table 34. STM32MP21xx OTP fuse description (lower OTP region)(1) (continued)

OTP words/ bits	Name/description	wplock(2)	Detailed description
8	IWDG3_FZ_STANDBY	-	IWDG3 freeze in Standby mode
9	IWDG4_HW	-	IWDG4 start on reset
10	IWDG4_FZ_STOP	-	IWDG4 freeze in Stop mode
11	IWDG4_FZ_STANDBY	-	IWDG4 freeze in Standby mode
12	SELINBORH	-	0: BOR Disabled 1: BOR = 2.7 V
13	Reserved for ST	-	-
14	HSLV_VDDIO3	-	V_{DDIO3} I/O segment below 2.5 V for I/O mode The I/O segment is used by OCTOSPI1
15	HSLV_VDDIO2	-	V_{DDIO2} I/O segment below 2.5 V for I/O mode The I/O segment is used by SDMMC2 port
16	HSLV_VDDIO1	-	V_{DDIO1} I/O segment below 2.5 V for I/O mode selection The I/O segment is used by SDMMC1 port
17	HSLV_VDD	-	Main I/O segment below 2.5 V for I/O mode
18	SYSRAM_ERASE	-	0: SYSRAM upper half not erased on reset 1: SYSRAM upper half erased on reset
19	Reserved for ST	-	0
20	DISABLE_SCAN	-	Scan chains are disabled on CLOSED_UNLOCKED device when this bit is 1
21 to 31	Reserved for ST	-	-
OTP125	MREPAIR1	Y	Reserved for ST
OTP126	MREPAIR2	Y	Reserved for ST
OTP127	MREPAIR3	Y	Reserved for ST

1. Refer to https://wiki.st.com/stm32mpu/wiki/STM32_MPU_ROM_code_overview.
2. Permanent programing locked.

Table 35. STM32MP21xx OTP fuse description (mid OTP region)

OTP words/ bits	Name/description	wplock(1)	Detailed description
OTP128	STM32CERTIF0(2)	Y	STM32 device certificate (public key), part1
OTP129	STM32CERTIF1	Y	STM32 device certificate (public key), part2
OTP130	STM32CERTIF2	Y	STM32 device certificate (public key), part3

Table 35. STM32MP21xx OTP fuse description (mid OTP region) (continued)

OTP words/ bits	Name/description	wplock(1)	Detailed description
OTP131	STM32CERTIF3	Y	STM32 device certificate (public key), part4
OTP132	STM32CERTIF4	Y	STM32 device certificate (public key), part5
OTP133	STM32CERTIF5	Y	STM32 device certificate (public key), part6
OTP134	STM32CERTIF6	Y	STM32 device certificate (public key), part7
OTP135	STM32CERTIF7	Y	STM32 device certificate (public key), part8
OTP136	STM32CERTIF8	Y	STM32 device certificate (public key), part9
OTP137	STM32CERTIF9	Y	STM32 device certificate (public key), part10
OTP138	STM32CERTIF10	Y	STM32 device certificate (public key), part11
OTP139	STM32CERTIF11	Y	STM32 device certificate (public key), part12
OTP140	STM32CERTIF12	Y	STM32 device certificate (public key), part13
OTP141	STM32CERTIF13	Y	STM32 device certificate (public key), part14
OTP142	STM32CERTIF14	Y	STM32 device certificate (public key), part15
OTP143	STM32CERTIF15	Y	STM32 device certificate (public key), part16
OTP144	STM32CERTIF16	-	STM32 device certificate (public key), part17
OTP145	STM32CERTIF17	-	STM32 device certificate (public key), part18
OTP146	STM32CERTIF18	-	STM32 device certificate (public key), part19
OTP147	STM32CERTIF19	-	STM32 device certificate (public key), part20
OTP148	STM32CERTIF20	-	STM32 device certificate (public key), part21
OTP149	STM32CERTIF21	-	STM32 device certificate (public key), part22
OTP150	STM32CERTIF22	-	STM32 device certificate (public key), part23
OTP151	STM32CERTIF23	-	STM32 device certificate (public key), part24
OTP152	OEM_KEY1_ROT0	-	OEM Key1 root of trust hash, part1
OTP153	OEM_KEY1_ROT1	-	OEM Key1 root of trust hash, part2
OTP154	OEM_KEY1_ROT2	-	OEM Key1 root of trust hash, part3
OTP155	OEM_KEY1_ROT3	-	OEM Key1 root of trust hash, part4
OTP156	OEM_KEY1_ROT4	-	OEM Key1 root of trust hash, part5
OTP157	OEM_KEY1_ROT5	-	OEM Key1 root of trust hash, part6
OTP158	OEM_KEY1_ROT6	-	OEM Key1 root of trust hash, part7
OTP159	OEM_KEY1_ROT7	-	OEM Key1 root of trust hash, part8
OTP160	OEM_KEY2_ROT0(2)	Y	OEM Key2 root of trust hash, part1
OTP161	OEM_KEY2_ROT1	Y	OEM Key2 root of trust hash, part2
OTP162	OEM_KEY2_ROT2	Y	OEM Key2 root of trust hash, part3
OTP163	OEM_KEY2_ROT3	Y	OEM Key2 root of trust hash, part4

Table 35. STM32MP21xx OTP fuse description (mid OTP region) (continued)

OTP words/ bits	Name/description	wplock(1)	Detailed description
OTP164	OEM_KEY2_ROT4	Y	OEM Key2 root of trust hash, part5
OTP165	OEM_KEY2_ROT5	Y	OEM Key2 root of trust hash, part6
OTP166	OEM_KEY2_ROT6	Y	OEM Key2 root of trust hash, part7
OTP167	OEM_KEY2_ROT7	Y	OEM Key2 root of trust hash, part8
OTP168	STM32PUBKEY0	Y	STM32 device public key, part1
OTP169	STM32PUBKEY1	Y	STM32 device public key, part2
OTP170	STM32PUBKEY2	Y	STM32 device public key, part3
OTP171	STM32PUBKEY3	Y	STM32 device public key, part4
OTP172	STM32PUBKEY4	Y	STM32 device public key, part5
OTP173	STM32PUBKEY5	Y	STM32 device public key, part6
OTP174	STM32PUBKEY6	Y	STM32 device public key, part7
OTP175	STM32PUBKEY7	Y	STM32 device public key, part8
OTP176	STM32PUBKEY8	Y	STM32 device public key, part9
OTP177	STM32PUBKEY9	Y	STM32 device public key, part10
OTP178	STM32PUBKEY10	Y	STM32 device public key, part11
OTP179	STM32PUBKEY11	Y	STM32 device public key, part12
OTP180 to OTP254	Available to customer	-	-
OTP255	ST_RSSE_EDMK_DERIV_CSTE_FUSE	Y	ST encryption decryption master key derivation constant

1.Permanent programing locked.
2.Key format 1.

Table 36. STM32MP21xx OTP fuse description (upper OTP region)

OTP words	Name/description	wplock(1)	Detailed description
OTP256	OTP_RMA_LOCK_PSWD0(2)	-	RMA_PSWD0
OTP257	OTP_RMA_LOCK_PSWD1	-	RMA_PSWD1
OTP258	OTP_RMA_LOCK_PSWD2	-	RMA_PSWD2
OTP259	OTP_RMA_LOCK_PSWD3	-	RMA_PSWD3

Table 36. STM32MP21xx OTP fuse description (upper OTP region) (continued)

OTP words	Name/description	wplock(1)	Detailed description
OTP260 to OTP347	OEM Secrets available to customer	-	-
OTP348	OEM_KEY2_EDMK0(2)	-	OEM master key used to derive FSBLM decryption key, part1
OTP349	OEM_KEY2_EDMK1	-	OEM master key used to derive FSBLM decryption key, part2
OTP350	OEM_KEY2_EDMK2	-	OEM master key used to derive FSBLM decryption key, part3
OTP351	OEM_KEY2_EDMK3	-	OEM master key used to derive FSBLM decryption key, part4
OTP352	OEM_KEY2_EDMK4		OEM master key used to derive FSBLM decryption key, part5
OTP353	OEM_KEY2_EDMK5		OEM master key used to derive FSBLM decryption key, part6
OTP354	OEM_KEY2_EDMK6		OEM master key used to derive FSBLM decryption key, part7
OTP355	OEM_KEY2_EDMK7		OEM master key used to derive FSBLM decryption key, part8
OTP356	OEM_KEY1_EDMK0		OEM master key used to derive FSBLA or M decryption key, part1
OTP357	OEM_KEY1_EDMK1		OEM master key used to derive FSBLA or M decryption key, part2
OTP358	OEM_KEY1_EDMK2		OEM master key used to derive FSBLA or M decryption key, part3
OTP359	OEM_KEY1_EDMK3		OEM master key used to derive FSBLA or M decryption key, part4
OTP360	OEM_KEY1_EDMK4	-	OEM master key used to derive FSBLA or M decryption key, part5
OTP361	OEM_KEY1_EDMK5	-	OEM master key used to derive FSBLA or M decryption key, part6
OTP362	OEM_KEY1_EDMK6	-	OEM master key used to derive FSBLA or M decryption key, part7
OTP363	OEM_KEY1_EDMK7	-	OEM master key used to derive FSBLA or M decryption key, part1
OTP364	STM32ENCPRVKEY0(3)	-	STM32 ECC device encrypted private key, part1
OTP365	STM32ENCPRVKEY1	-	STM32 ECC device encrypted private key, part2
OTP366	STM32ENCPRVKEY2	-	STM32 ECC device encrypted private key, part3
OTP367	STM32ENCPRVKEY3	-	STM32 ECC device encrypted private key, part4
OTP368	STM32ENCPRVKEY4	-	STM32 ECC device encrypted private key, part5
OTP369	STM32ENCPRVKEY5	-	STM32 ECC device encrypted private key, part6
OTP370	STM32ENCPRVKEY6	-	STM32 ECC device encrypted private key, part7
OTP371	STM32ENCPRVKEY7	-	STM32 ECC device encrypted private key, part8
OTP372	STM32ENCPRVKEY8	-	STM32 ECC device encrypted private key, part9
OTP373	STM32ENCPRVKEY9	-	STM32 ECC device encrypted private key, part10

Table 36. STM32MP21xx OTP fuse description (upper OTP region) (continued)

OTP words	Name/description	wplock(1)	Detailed description
OTP374	STM32ENCPRVKEY10	-	STM32 ECC device encrypted private key, part11
OTP375	STM32ENCPRVKEY11	-	STM32 ECC device encrypted private key, part12
OTP376	HWKEY0	-	Secret hardware unique key, part1
OTP377	HWKEY1	-	Secret hardware unique key, part2
OTP378	HWKEY2	-	Secret hardware unique key, part3
OTP379	HWKEY3	-	Secret hardware unique key, part4
OTP380	HWKEY4	-	Secret hardware unique key, part5
OTP381	HWKEY5	-	Secret hardware unique key, part6
OTP382	HWKEY6	-	Secret hardware unique key, part7
OTP383	HWKEY7	-	Secret hardware unique key, part8

1.Permanent programing locked.
2.Key format 1
3.Key format 3

8 Resource isolation framework security controller (RIFSC)

8.1 Introduction

Resource isolation framework (RIF) is a comprehensive set of hardware blocks designed to enforce and manage isolation of STM32 hardware resources, like memory and peripherals.

The RIFSC (RIF security controller) is composed of three sets of registers, each managing a specific set of hardware resources:

- **RISC** registers associated with **RISUP** logic (resource isolation slave unit for peripherals), assign all non RIF-aware peripherals to zero, one or any security domains (secure, privilege, compartment).
- **RIMC** registers: associated with **RIMU** logic (resource isolation master unit for address space protection), assign all non RIF-aware bus master to one security domain by setting secure, privileged, and compartment information on the system bus. Alternatively, the RISUP logic controlling the slave port access to a peripheral can assign target bus attributes to this peripheral master port (supported attribute: CID).

8.2 RIFSC main features

- As a RIF core component, assign specific device resources to security domains: isolating resources defined as peripherals and some variable sized address space regions
- Centralized set of registers controlling the RIF logic associated to legacy slave peripheral targets: RISUP
- Centralized set of registers controlling the RIF logic associated to master ports of legacy peripherals: RIMU
- Support for legacy master port RIF attribute inheritance from configuration port programming (supported attribute: CID)
- Secure guard: legacy master port forced to non-secure when its associated configuration port is programmed non-secure
- Built-in hardware semaphores to change dynamically ownership of peripheral resources
- Each compartment CPU can see only the interrupt events of the legacy peripheral resources allocated to it
- Support to RIF special debug domain, with greater accesses for debugger used in secure or non-secure global debug sessions

8.3 RIFSC implementation

RIMU

Master port RIF protections in this device (using RIMU) are defined in *Table 37*. Possible inheritance from RISUP of RIF configuration is also defined.

Table 37. RIMU resource assignment

Index(1)	Master name	CID at reset	RISUP(2) inheritance
-	CPU1 (Cortex-A35)	0x1	No
-	CPU2 (Cortex-M33)	0x2	
0	Trace (ETR)	0x0	
1	SDMMC1	RISUP76	CID, secure guard(3)
2	SDMMC2	RISUP77	
3	SDMMC3	RISUP78	
4	OTG_HS	RISUP66	
5	USBH	RISUP63	
6	ETH1	RISUP60	
7	ETH2	RISUP61	
10	DCMIPP (camera)	RISUP87	
11	LTDC L1 or L2	0x0	No
12	LTDC L3		

1. When no index is written, the CID values are hardcoded, and the master manages security.
2. When applicable, the corresponding CIDSEL bit is available in RIFSC_RIMC_ATTRx.
3. The master is forced by hardware to non-secure when its associated RISUP is programmed non-secure.

Note: All DMA instances (HPDMA, LPDMA) are not mentioned, as they automatically manage the CID to use on their master ports (0 to 2, or 0 to 3 only), inheriting the value from the CPU that controls a given channel.

RISUP

Table 38 defines the index of the non-RIF-aware peripherals managed by the RIFSC (RISUP blocks) and the IAC. Few RIF-aware peripherals are also listed.

When a peripheral is used by multiple security domains (like multiple CPU codes), the RCC implements specific clock and reset access control registers, using a dedicated *rcc* index, also shown in this table.

Table 38. Indexes for non-RIF-aware peripherals (RISUP)

Peripherals 31 to 0																															
-	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	USART1	SPDIFRX	-	-	SPI6	SPI5	SPI4	SPI3	SPI2	SPI1	LPTIM5	LPTIM4	LPTIM3	LPTIM2	LPTIM1	-	TIM17	TIM16	TIM15	TIM14	TIM13	TIM12	TIM11	TIM10	TIM8	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2	TIM1
Peripherals 63 to 32																															
-	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	USBH	-	ETH2	ETH1	ADC2	ADC1	HDP	FDCAN	-	MDF1	-	SAI4	SAI3	SAI2	SAI1	-	-	-	-	-	I2C3	I2C2	I2C1	LPUART1	-	-	UART7	USART6	UART5	UART4	USART3	USART2
Peripherals 95 to 64																															
-	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
	SAES	PKA	RNG2	RNG1	-	-	-	DCMI PSSI	DCMIPP	CSI	-	-	-	-	-	LTDC common(1)	-	SDMMC3	SDMMC2	SDMMC1	-	OCTOSPI1	STGEN(2)	-	-	-	-	-	DDRPERFM	OTG_HS	-	-
rcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	105	-	-	-	-	-	110	-	-	-	-	-	-	-	-	-	-
Peripherals 127 to 96																															
-	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
	IAC(2)(3)	-	OTFDEC1	-	-	-	-	LTDC_L3(3)	LTDC_L1L2(3)	ICACHE DCACHE(3)	-	I3C3	I3C2	I3C1	-	-	-	SERC	CRC	RAMCFG	DTS	VREFBUF	-	WWDG1	IWDG4	IWDG3	IWDG2	IWDG1	CRYP2	CRYP1	HASH2	HASH1
rcc	-	-	110	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Common configuration registers (refer to the corresponding peripheral section for details).
2. RIF-aware peripheral, without access control registers in the RIFSC.
3. No associated registers in the RCC.

IAC and RCC indexes for RIF-aware peripherals and RIF protected memories

Table 39 defines the index of the RIF firewall components and RIF-aware peripherals managed by IAC and RCC peripherals.

Note: All these peripherals are RIF-aware, not requiring RISUP, and usually belonging to multiple compartments.

Table 39. Indexes for RIF-aware peripherals (with RCC control index)

Peripherals 159 to 128																																
-	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
	SYSCFG	RIFSC	CA35SSC	RCC	PWR	EXTI2	EXTI1	TAMP	RTC	-	HPDMA3	HPDMA2	HPDMA1	-	-	IPCC1	-	-	-	STGEN	-	-	RISAF4 (DDR-SDRAM)	-	RISAF2 (OCTOSPI1)	RISAF1 (BKPSRAM)	-	RISAB5 (RETRAM)	-	RISAB3 (SRAM1)	RISAB2 (SYSRAM high)	RISAB1 (SYSRAM low)
rcc	-	-	106	-	-	-	-	102	102	-	85	84	83	-	-	87	-	-	-	-	-	-	104	-	110/111	77	-	76	-	78	74	74
Peripherals 191 to 160																																
-	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	BSEC	FMC	DDRPHYC	DDRCTRL	-	-	GPIOZ	-	-	GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
rcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	103	112	104	104	-	-	101	-	-	98	97	96	95	94	93	92	91	90

RIFSC, RCC, and IAC common programming

To program the RISUP and the RCC, and to read back illegal access information in the IAC, refer to *Section 5.4: Resource isolation framework (RIF)*.

8.4 RIFSC functional description

8.4.1 RIFSC block diagrams

The RIFSC is composed of two sets of registers, each managing a specific set of hardware resources: RIMU and RISUP

RIFSC and RIMU block diagram

The following figures show the RIMU block diagrams, associated with its master on the interconnect, and the RIFSC peripheral. RISUP inheritance is defined in *Section 8.3*.

Figure 18. RIMU block diagram (without inheritance)

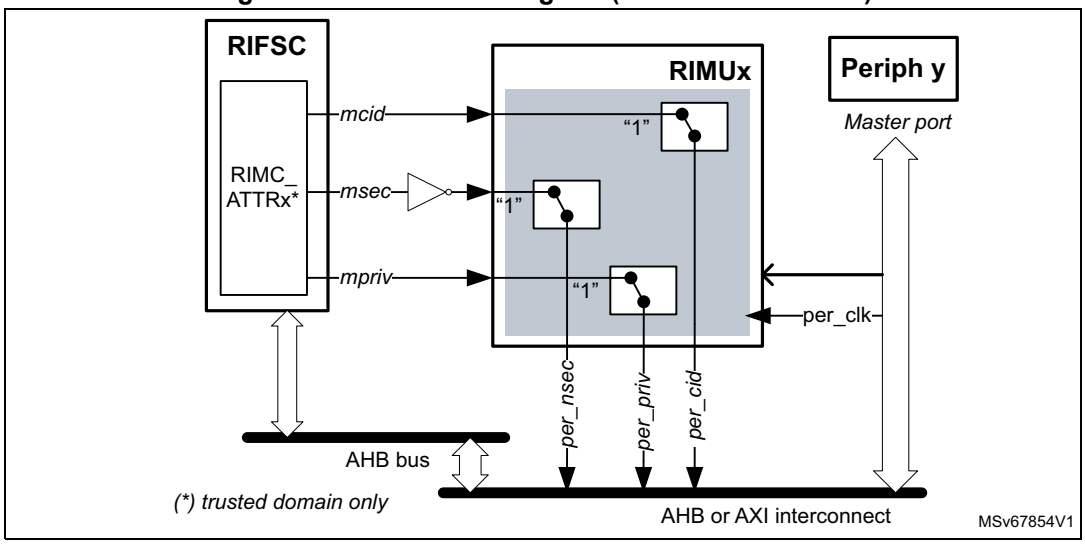
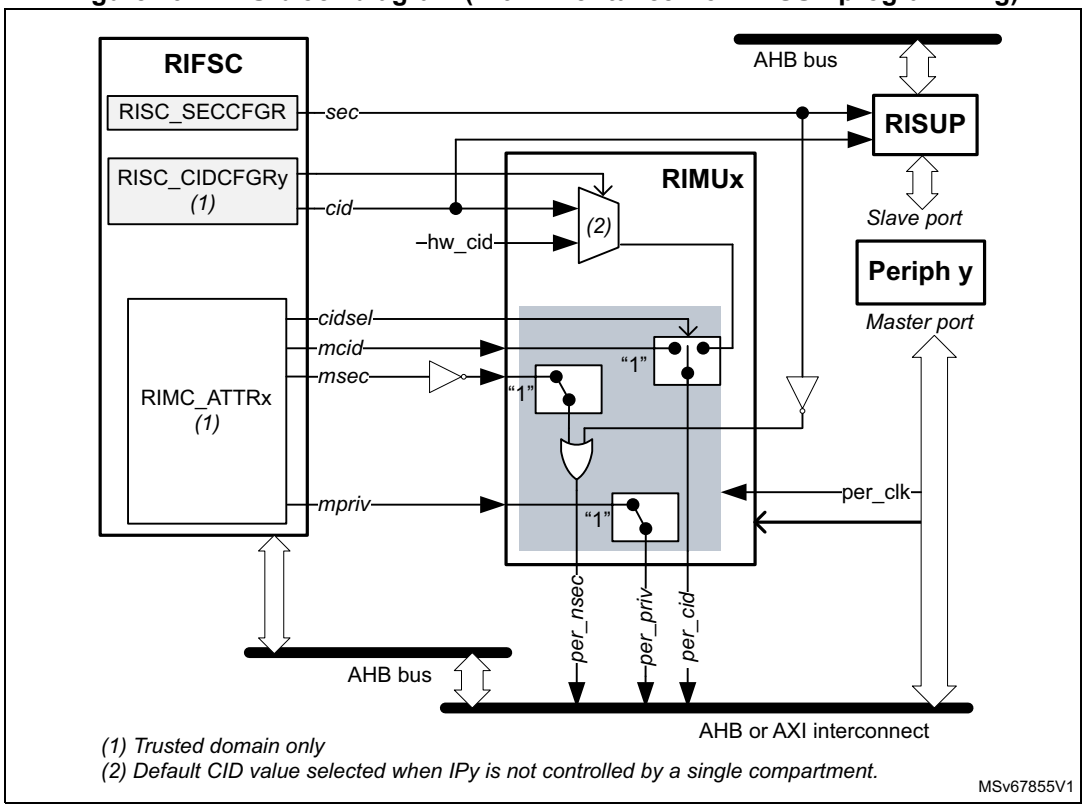


Figure 19. RIMU block diagram (with inheritance from RISUP programming)

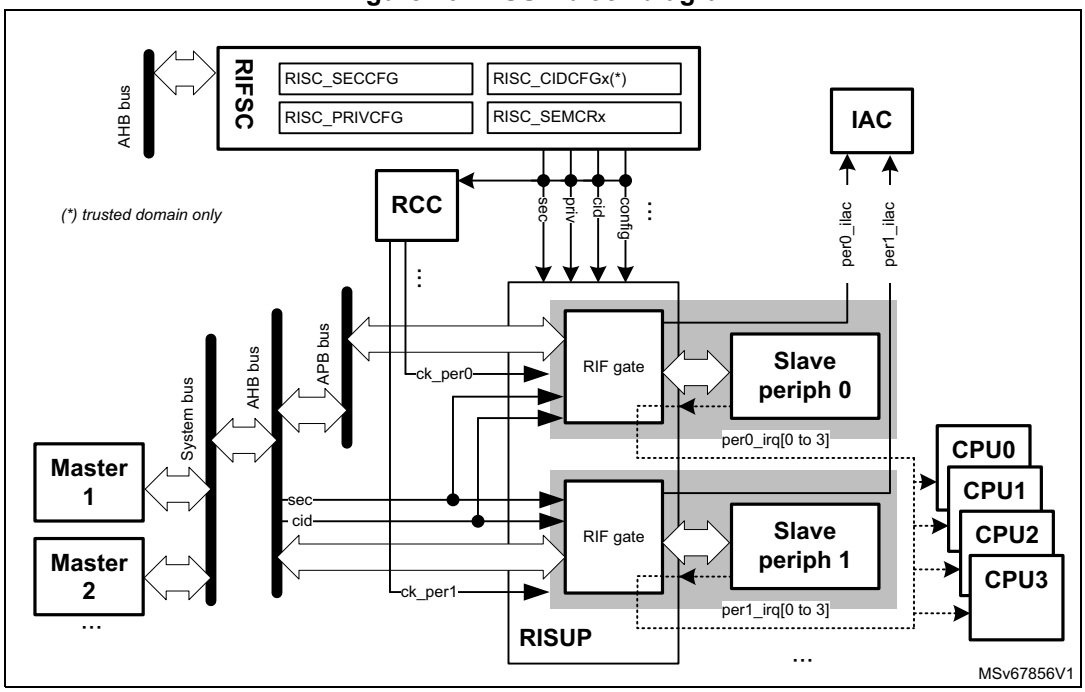


RIFSC and RISUP block diagram

Figure 20 shows the block diagram of the RISUP, associated with its slaves on the interconnect, the IAC, and the RIFSC peripherals.

The RISUP logic protecting a peripheral uses information from the RIFSC to determine which security, privilege, and compartment are allowed to go through. There is also inheritance between the access right to the peripheral and the access to the corresponding clocking and reset bits in the RCC peripheral.

Figure 20. RISUP block diagram



8.4.2 RIFSC internal signals

Table 40 and *Table 41* describe some internal signals available at, respectively, RIMU and RISUP logic levels.

Table 40. RIMU internal input/output signal

Name	Type	Description
per_clk	Digital input	Peripheral master port bus clock

Table 41. RISUP internal input/output signals

Name	Type	Description
ck_perX	Digital input	Peripheral x kernel clock from RCC
ilac	Digital output	Illegal access signal to IAC

8.4.3 RIFSC reset and clocks

RIMU and RISUP logics share the same clock and the same reset as the resource they are protecting.

The RIFSC has a single reset. After a SoC general reset (power-on reset), all non RIF-aware peripherals become non-secure and unprivileged, reachable by any CID.

8.4.4 RIFSC control of masters (RIMU)

RIMC registers in RIFSC, associated with the RIMU logic, assign bus master x to a security domain by setting secure, privileged, and compartment information on the system bus using RIMC_ATTRx registers. Alternatively, when clearing the corresponding SEL bit in RIMC_ATTRx, the bus attribute of the master port x is taken from the RISUP that controls the access to slave port of this peripheral x (supported attribute: CID).

RIMC registers must be initialized by a trusted domain application after the device reset. Optionally this application can set the GLOCK bit in RIMC_CR, preventing further writes to all RIMC registers. This bit can be cleared only by a power-on reset.

If a trusted domain application programs a compartment ID equals 4, 5, or 6 using the MCID[2:0] bitfield, these masters are usually sand boxed to dedicated chunk of memories, and have a reduced set of peripherals and GPIOs.

Note: In the STM32MP21xx device, CID 7 is reserved for multicore debugging (see Section 5.4.7: RIF special security domain: debug domain).

8.4.5 RIFSC control of debug access port (DAP)

For the protection of the DAP master port, a specific hardware has been implemented in the device. The table below summarizes how the DAP peripheral performs its accesses on the interconnect, based on the debug profile selected by the secure application in the device.

Table 42. DAP compartment ID allocation

Debug profile	CPU debug	MCUs debug	DAP CID value	Comment
Microcontroller debug	No	Yes	same as Cortex-M33 (0x2)	In this profile, both MCUs can be debugged. Only peripherals accessible to Cortex-M33 can be accessed by the DAP, if security and privilege allow it.
Application processor debug	Yes	No	same as Cortex-A35 (0x1)	Only peripherals accessible to Cortex-A35 can be accessed by the DAP, if security and privilege allow it.
Global debug (MCUs and CPU)	Yes	Yes	Any CID (default: 0x7)	DAP CID is defined by DAPCID bitfield in RIMC_CR. DAPCID is writable by secure, non-secure, privileged or unprivileged DAP.

Three important notes are linked to this table:

- In **microcontroller** and **application processor debug** modes, the DAP has exactly the same right in the RIF as the Cortex core under debug (Cortex-M or Cortex-A).
- In **non-secure global debug mode,** when DAPCID = 0x7, the DAP in privileged mode can access (in read and write) all non-secure resources without changing the RIF configuration.
- In **secure global debug mode,** when DAPCID = 0x7, the DAP in privileged mode can access (in read and write) any resources in the device (peripheral and memory), including registers reserved to the trusted domain.

The way to set up these debug profiles in the device is explained in *Section 6.3.11: BSEC debug and trace control*.

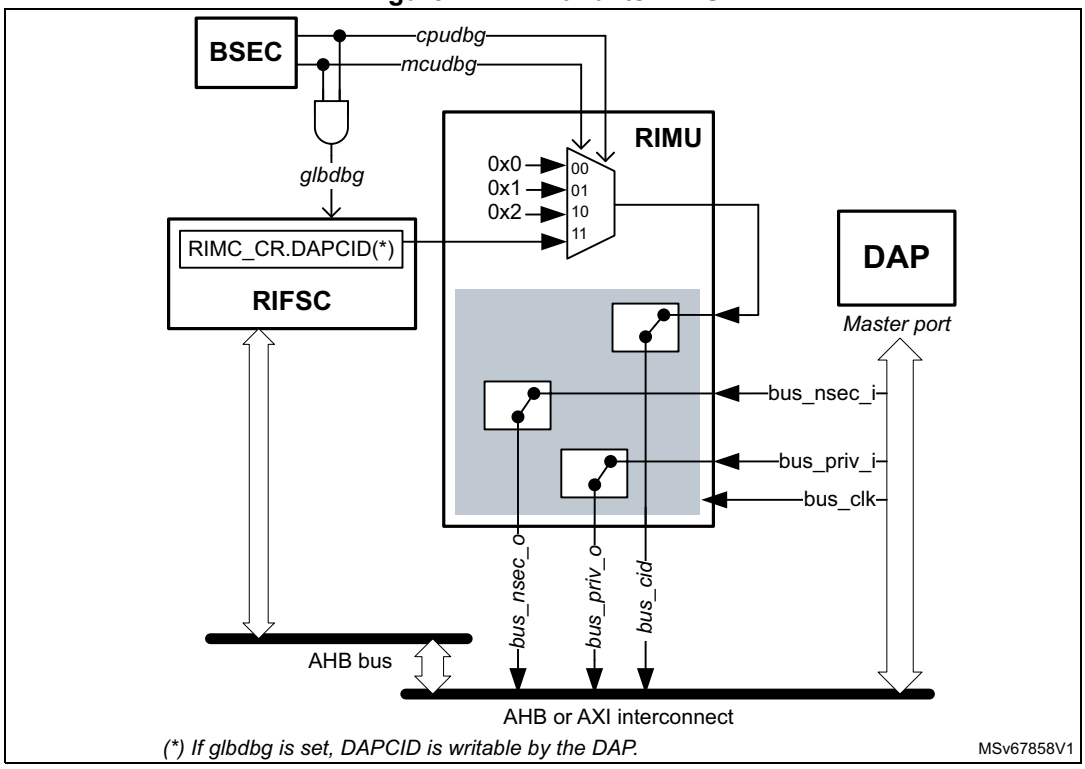
Figure 21 illustrates how BSEC, DAP, RIMU, and RIMC are integrated. If a global debug profile is selected in BSEC (CPU + MCU debug activated), the DAPCID bitfield in RIMC_CR is writable by secure, non-secure, privileged, or unprivileged DAP. The GDPEN bit is also set in RIMC_SR. If any other debug profile is selected, RIMC_CR can be written only by a trusted domain application.

Note: When no debug profile is selected, the DAP CID is zero.

When DAPCID is left with its default value (0x7), the DAP is assigned to the RIF debug domain, when a global debug profile is selected.

DAP privileged level is not defined by RIMU. Hence, if debug tools define the DAP as unprivileged, its access can be rejected by the target even if DAPCID = 0x7.

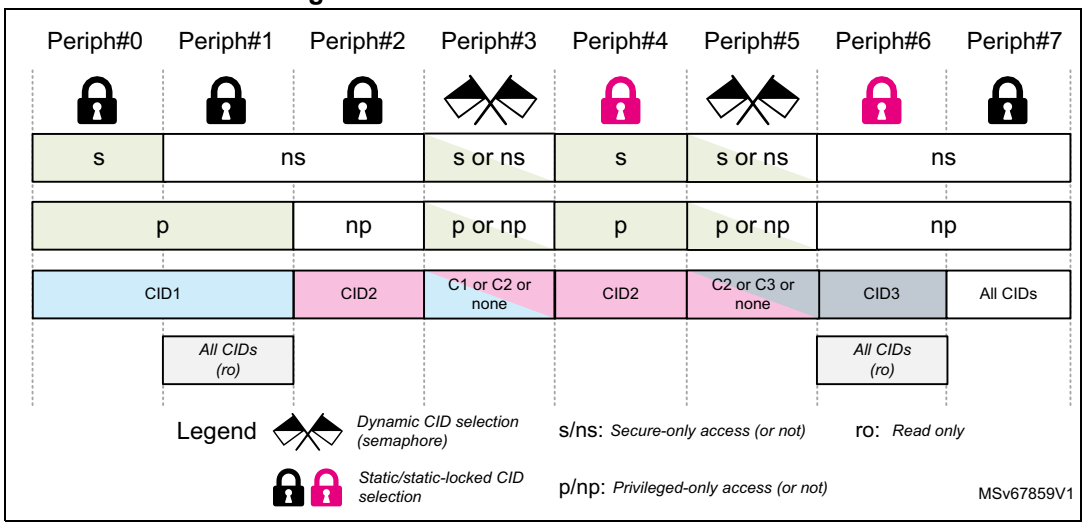
Figure 21. DAP and its RIMU



8.4.6 RIFSC control of peripherals (RISUP)

RISC registers in RIFSC, associated with the RISUP logic, assign non-RIF aware peripherals to none, one or any security domain (secure, privilege, compartment). The figure below illustrates the principle of this allocation.

Figure 22. Allocation of RISUP resources



For each peripheral, the RIF allows the following policies:

- any compartment access (default configuration)
- single-compartment access, statically defined by the trusted domain application
- single-compartment access, dynamically taken by non-trusted domain applications, granted by the trusted domain
- no compartment access, statically configured by the trusted domain application

Note: For the first three cases, the filtering based on secure and privileged always applies.

At any point in time, the trusted domain application can lock the RIF configuration of a peripheral until the next reset, by setting the RLOCK bit in RISC_RCFGLOCKR.

As defined in *Section 5.4: Resource isolation framework (RIF)*, the debug domain (privileged) must be granted following accesses when activated:

- A secure debug domain can read and write anywhere in the device, including registers reserved for the trusted domain.
- A non-secure debug domain can only access (in read and write) peripherals that are tagged by the application as non-secure.

RISUP application information

To assign statically the peripheral x to the compartment y, a trusted domain application must set SCID to CIDy and CFEN to 1 in RISC_PERx_CIDCFGR. An application in compartment y can then set or clear the SECx bit in RISC_SECCFG, and the PRIVx bit in RISC_PRIVCFGR, if the security and privilege of the access are sufficient.

To grant statically the peripheral x to any compartment, a trusted domain application must clear CFEN bit to 0 in RISC_PERx_CIDCFGR. An application in any compartment can then set or clear SECx in RISC_SECCFG, and PRIVx in RISC_PRIVCFGR, if the security and privilege of the access are sufficient.

Note: Only a secure privileged application can write to RISC_SECCFGR, while only a privileged application can write to RISC_PRIVCFGR.

The SECx bit in RISC_SECCFG controls the secure access to PRIVx in RISC_PRIVCFGR.

The dynamic assignment of peripheral x to compartment y1 or y2 is done as follows:

- A trusted domain application sets the white-listed SEMWLCy1 and SEMWLCy2 bits in RISC_PERx_CIDCFGR. The SEM_EN and CFEN bits must also be set.
- An application running in Cy1 or Cy2 compartment can now take the SEM_MUTEX bit in RISC_PERx_SEMCR, by setting it to 1. When SEM_MUTEX = 0, the first white-listed application that takes this bit owns it (only this application can clear the MUTEX bit.
- When the RISUP semaphore takes the peripheral x and associated clocking, the reset bits in the RCC peripheral become accessible to the application that has taken the semaphore. This application can then set or clear SECx in RISC_SECCFG, and PRIVx in RISC_PRIVCFGR, if the security and privilege of the access are sufficient. If the applications running in other compartments try to access this peripheral, the reads return zero and the writes are ignored.

Note: At any time, a trusted domain application can reset the peripheral x semaphore by clearing then setting SEM_EN in RISC_PERx_CIDCFGR.

When the trusted domain needs to prevent any compartment to access a peripheral x, the recommended procedure is the following:

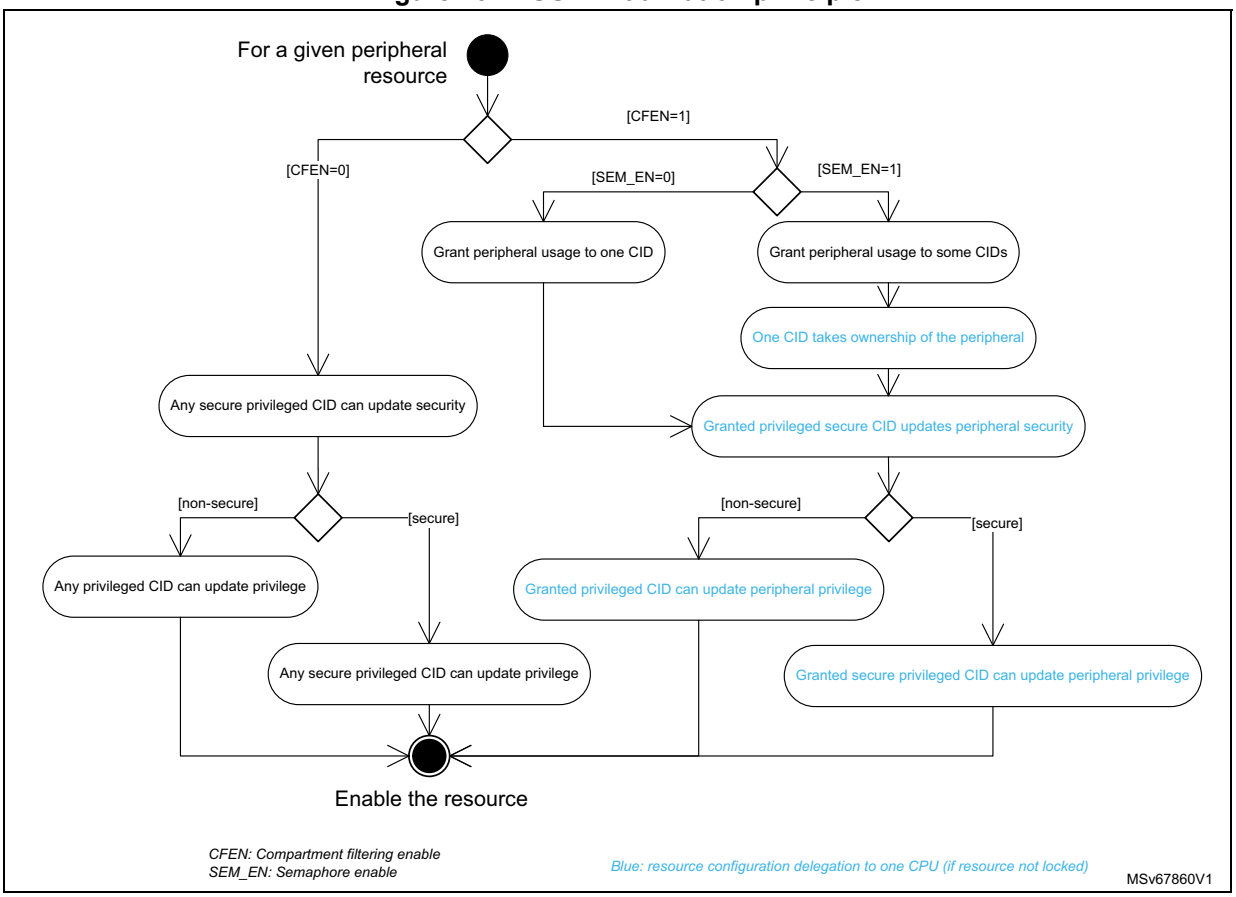
1. Mask illegal access interrupts from peripheral x in IAC_IER register.
2. Set RISC_CIDCFGRx to statically assign it to TDCID.
3. Set RCC_xCFGR with default values.
4. Reset all SEMWLCy bits, and set SEM_EN and CFEN bits in RISC_PERx_CIDCFGR.

From that point, no applications can take the semaphore, hence the access to the peripheral is prevented.

Note: Steps 2 to 3 are not needed if the RCC bits corresponding to this peripheral are at the default values.

The above methods are summarized in the figure below.

Figure 23. RISUP initialization principle



8.4.7 Managing illegal accesses

RIF component RISUP does not use the standard interrupt lines toward NVIC or GIC concentrators. The IAC captures all RIF interrupt events, called illegal access events.

Refer to *Section 11: Illegal access controller (IAC)* for more information on the way illegal accesses are managed.

8.5 RIFSC registers

These registers are accessible through the register interface of the RIFSC peripheral.

Registers *Section 8.5.1* to *Section 8.5.6* are slave control registers for peripherals.

Registers *Section 8.5.7* and *Section 8.5.9* are master configuration registers.

If security and privileged policy allows it, the debug domain traffic goes through, regardless of any CID filtering policy.

8.5.1 RIFSC RISC slave configuration register x (RIFSC_RISC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged, trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Global lock

This bit locks the configuration of RIFSC RISC registers until next reset. This bit is cleared by default and, once set, it cannot be reset until global RIFSC reset.

0: RIFSC RISC registers are writable.

1: All writes to RIFSC RISC registers are ignored, except RIFSC_RISC_PERy_SEMCR.

8.5.2 RIFSC RISC slave security configuration register x (RIFSC_RISC_SECCFGRx)

Address offset: 0x010 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Non-secure or unprivileged writes to this register are ignored, while any read is allowed. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC{i + 32 * x}**: Security configuration for peripheral {i + 32 * x} (i = 0 to 31)

0: Secure and non-secure data access are granted to the peripheral {i + 32 * x}.

1: Secure data access only are granted to the peripheral {i + 32 * x}.

*Note: Regardless of SEC value, any execute access to peripheral {i + 32 * x} triggers a bus error.*

*If the compartment filtering for peripheral {i + 32 * x} is enabled (resp. disabled) using CFEN in RIFSC_RISC_PERy_CIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if privilege and security of the access is sufficient.*

Authorized compartment is defined as SCID if SEM_EN = 0 in RIFSC_RISC_PERy_CIDCFGR. If SEM_EN = 1, it is equal to SEMCID, if SEM_MUTEX = 1 in RIFSC_RISC_PERy_SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

8.5.3 RIFSC RISFC slave privileged register x (RIFSC_RISC_PRIVCFGRx)

Address offset: 0x030 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, while any read is allowed. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV{i + 32 * x}**: Privileged-only access permission for peripheral {i + 32 * x} (i = 0 to 31)

0: Privileged and unprivileged data access are granted to the peripheral {i + 32 * x}.

1: Privileged data access only are granted to the peripheral {i + 32 * x}.

*Note: If corresponding SEC{i + 32 * x} bit is set in RIFSC_RISC_SECCFGRx, this bit can only be written by a secure privileged application.*

*If the compartment filtering for peripheral {i + 32 * x} is enabled (resp. disabled) using CFEN in RIFSC_RISC_PERy_CIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if privilege and security of the access is sufficient.*

Authorized compartment is defined as SCID if SEM_EN = 0 in RIFSC_RISC_PERy_CIDCFGR. If SEM_EN = 1, it is equal to SEMCID, if SEM_MUTEX = 1 in RIFSC_RISC_PERy_SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

8.5.4 RIFSC RISC slave resource configuration lock register x (RIFSC_RISC_RCFGLOCKRx)

Address offset: 0x050 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **RLOCK{i + 32 * x}**: rRsource lock for peripheral {i + 32 * x} (i = 0 to 31)
This bit is set to lock the peripheral resource {i + 32 * x}. It is cleared by default and, once set, this bit cannot be cleared until the RIFSC RISC peripheral is reset.
0: SEC{i + 32 * x} in RIFSC_RISC_SECCFGRx and PRIV{i + 32 * x} in RIFSC_RISC_PRIVCFGRx are writable. RIFSC_RISC_PERy_CIDCFGR is writable.
1: Writes to SEC{i + 32 * x} and PRIV{i + 32 * x} are ignored. Writes to RIFSC_RISC_PERy_CIDCFGR register are ignored.

8.5.5 RIFSC RISC slave peripheral y CID configuration register (RIFSC_RISC_PERy_CIDCFGR)

Address offset: 0x100 + 0x8 * y, (y = 0 to 127)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RIFSC_RISC_CR, or if RLOCKy is set in RIFSC_RISC_RCFGLOCKRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
									rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SEMWLCi**: Semaphore white list for compartment i (i = 7 to 0)
0: Application running in compartment i cannot get access to peripheral y through the semaphore SEM_MUTEX.
1: Application running in compartment i can get access to peripheral y through the semaphore SEM_MUTEX.
Note: This bit is not taken into account by RISC when the semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **SCID[2:0]**: Static CID
The trusted domain application uses this bitfield to statically define which compartment has access to the peripheral y when SEM_EN = 0.
Note: This CID is not taken into account by RISC when the semaphore is enabled (SEM_EN = 1) or compartment filtering is off (CFEN = 0).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEM_EN**: Semaphore enable
0: Access to peripheral y is granted only if CID of the access equals to SCID.
1: Access to peripheral y is granted only if SEM_MUTEX = 1 and CID of the access equals to SEMCID in RIFSC_RISC_PERy_SEMCR.
Note: This bit is not taken into account by RISC if the compartment filtering is off (CFEN = 0). When this bit is cleared from 1 by the trusted domain application, SEM_MUTEX is cleared to zero by RISC.

Bit 0 **CFEN**: CID filtering enable

0: Compartment-based filtering defined in RIFSC_RISC_PERy_CIDCFGR and optionally RIFSC_RISC_PERy_SEMCR is disabled for peripheral y.

1: Compartment-based filtering defined in RIFSC_RISC_PERy_CIDCFGR and optionally RIFSC_RISC_PERy_SEMCR is enabled for peripheral y.

For any CFEN value, access controls to peripheral y defined in RIFSC_RISC_PRIVCFGRx and RIFSC_RISC_SECCFGRx always apply.

Note: When this bit is cleared from 1 by the trusted domain application, SEM_MUTEX is cleared to zero by RISC.

8.5.6 RIFSC RISC slave peripheral y semaphore control register (RIFSC_RISC_PERy_SEMCR)

Address offset: 0x104 + 0x8 * y, (y = 0 to 127)

Reset value: 0x0000 0000

If the compartment filtering is disabled (CFEN = 0) or the semaphore is disabled (SEM_EN = 0) in RIFSC_RISC_PERy_CIDCFGR, any write to this register is ignored.

Privileged-only write if PRIVy is set in RIFSC_RISC_PRIVCFGRx. Secure-only write if SECy bit is set in RIFSC_RISC_SECCFGRx.

If compartment filtering is enabled and semaphore is enabled (CFEN = SEM_EN = 1), only an authorized compartment can write to this register. For this register, the authorized compartment is equal to SEMCID if SEM_MUTEX = 1, else it is defined by the CID white list (SEMWLCy) in register RIFSC_RISC_PERy_CIDCFGR. There is no authorized compartment if the semaphore is disabled (SEM_EN = 0).

Any read access is allowed to this register. The debug domain is managed like any other masters, when activated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_ MUTEX
									r	r	r				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SEMCID[2:0]**: Semaphore current CID

When SEM_EN and CFEN are set in RIFSC_RISC_PERy_CIDCFGR, this read-only bitfield is updated with the CID value of the latest successful acquisition of RIFSC_RISC_PERy_SEMCR semaphore (SEM_MUTEX = 1).

In all the other cases, this bitfield stays unchanged.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX**: Semaphore mutex

A white-listed application can set this bit to get exclusive access to peripheral y when SEM_EN is set in RIFSC_RISC_PERy_CIDCFGR (writes are ignored otherwise).

Read 0: Semaphore is free to take.

Read 1: Semaphore is taken.

Write 0: Release semaphore.

Write 1: Acquire semaphore.

Note: When SEM_MUTEX = 0, only applications belonging to white-listed compartments (defined with SEMWLCi bits in RIFSC_RISC_PERy_CIDCFGR) can acquire this semaphore.

When SEM_MUTEX = 1, only applications belonging to compartment SEMCID can release this semaphore.

This bit is cleared when CFEN or SEM_EN are cleared from 1 by the trusted domain application in RIFSC_RISC_PERy_CIDCFGR.

8.5.7 RIFSC RIMC master configuration register (RIFSC_RIMC_CR)

Address offset: 0xC00

Reset value: 0x0000 8710

Any read access is allowed on this register. Writes are ignored if GLOCK is set in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDEN	Res.	Res.	Res.	Res.	DAPCID[2:0]			Res.	TDCID[2:0]			Res.	Res.	WUCD ONE	GLOCK
r					rw	rw	rw		rwo	rwo	rwo			rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DDEN**: Debug domain enable

This read-only bit gives the status on the debug domain feature availability on this device.

0: Debug domain not available

1: Debug domain available

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **DAPCID[2:0]**: Debug access port compartment ID

This bitfield defines the CID of the DAP when GDPEN is set in RIFSC_RIMC_SR.

When DAPCID[2:0] = 0x7, debugging tools have the highest level of debug access, if secure global debug profile is enabled in the BSEC peripheral.

When GDPEN = 0, DAPCID[2:0] can be written by trusted application (TDCID secure privileged). When GDPEN = 1, the DAP can also change this bitfield regardless of the access type (secure, non-secure, privileged, or unprivileged).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **TDCID[2:0]**: Trusted domain compartment ID

Master with this compartment ID runs the application with the highest level of trust. TDCID[2:0] value is distributed in all the device. This bitfield is write once (subsequent writes are ignored).

Note: Secure privileged write access only. Write only by debug domain and trusted domain CIDs, as defined in this register.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **WUCDONE**: Wake-up CPU done

Setting this bit deactivates the wake-up CPU mechanism described in *Section 5.4: Resource isolation framework (RIF)*. Reading this bit always returns zero.

0: When WUCEN = 1 in RIFSC_RIMC_SR, the trusted domain CID information is fixed to 0x2 (wake-up CPU), else it is defined in this register.

1: Setting this bit clears WUCEN after few cycles. Once WUCEN =0, the trusted domain CID information is defined in this register.

Note: Secure privileged write access only. Write only by debug domain and trusted domain CIDs, as defined in this register (including this bit).

Bit 0 **GLOCK**: Global lock

This bit is used to lock the configuration of RIFSC RIMC registers until next reset. This bit is cleared by default and, once set, it cannot be reset until global RIFSC reset.

0: RIFSC RIMC registers are writable.

1: All writes to RIFSC RIMC registers are ignored.

Note: Secure privileged write access only. Write only by debug domain and trusted domain CIDs, as defined in this register.

8.5.8 RIFSC RIMC master status register (RIFSC_RIMC_SR)

Address offset: 0xC04

Reset value: 0x0000 0000

Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCEN	GDPEN
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WUCEN**: Wake-up CPU enable

This read-only bit is set when the application properly initializes the wake-up CPU mechanism described in *Section 5.4: Resource isolation framework (RIF)*.

0: Wake-up CPU is disabled. The trusted domain CID value is driven by RIFSC_RIMC_CR.

1: Wake-up CPU is enabled. The trusted domain CID value is fixed to 0x2 until the application sets WUCDONE in RIFSC_RIMC_CR (in this case, WUCDONE is cleared after few cycles).

Bit 0 **GDPEN**: Global debug profile enable

This read-only bit is set when the global-debug profile is activated in the device (CPU and MCU debugging allowed).

0: Global-debug profile (CPU + MCU) is not activated.

1: Global-debug profile (CPU + MCU) is activated.

8.5.9 RIFSC RIMC master attribute register x (RIFSC_RIMC_ATTRx)

Address offset: 0xC10 + x * 0x4 (x = 0 to 15)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RIFSC_RIMC_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MPRIV	MSEC	Res.	MCID[2:0]			Res.	CIDSEL	Res.	Res.
						rw	rw		rw	rw	rw		rw		

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MPRIV**: Master privileged

Value of the privileged flag on the interconnect for this master, programmed by the trusted domain.

0: This master is unprivileged.

1: This master is privileged.

Bit 8 **MSEC**: Master secure

Value of the secure flag on the interconnect for this master, programmed by the trusted domain.

0: This master is non-secure.

1: This master is secure.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **MCID[2:0]**: Master CID

Value of the CID flag on the interconnect for this master, programmed by the trusted domain. Valid only if indicated in RIF masters table in *Section 8.3*. This bitfield is not taken into account when CIDSEL = 0 in this register.

This bitfield cannot be written with a value of 0x7 (write to MCID[2:0] is ignored).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CIDSEL**: CID selection

0: Compartment ID of master x is driven by hardware value or by inheritance to associated RISUP. See master table in *Section 8.3* for details.

1: Compartment ID of master x is defined by MCID[2:0] in this register.

Bits 1:0 Reserved, must be kept at reset value.

8.5.10 RIFSC peripheral protection status register x (RIFSC_PPSRx)

Address offset: 0xFB0 + 0x4 * x, (x = 0 to 3)

Reset value: 0xCFFE FFFF, 0xBFDE 0F3F, 0xF9C3 740C, 0x21DC 7DFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 32 * x}**: Peripheral protection enable {i + 32 * x} (i = 0 to 31)

0: RISUP protection to peripheral {i + 32 * x} is not present. The corresponding RIFSC_RISC_PERy_CIDCFGR and RIFSC_RISC_PERy_SEMCR (where y = i + 32 * x) are reserved.

1: RISUP protection to peripheral {i + 32 * x} is present.

8.5.11 RIFSC hardware configuration register 3 (RIFSC_HWCFGR3)

Address offset: 0xFE8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CFG[31:0]**: Hardware configuration

This bitfield returns the generic value RISAL_FIXED_LENGTH.

8.5.12 RIFSC hardware configuration register 2 (RIFSC_HWCFGR2)

Address offset: 0xFEC

Reset value: 0x0010 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG3[7:0]								CFG2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **CFG3[7:0]**: Hardware configuration 3

This bitfield returns the generic value NUM_RISAL.

Bits 23:16 **CFG2[7:0]**: Hardware configuration 2
This bitfield returns the generic value NUM_MP.

Bits 15:0 **CFG1[15:0]**: Hardware configuration 1
This bitfield returns the generic value NUM_SP.

8.5.13 RIFSC hardware configuration register 1 (RIFSC_HWCFGR1)

Address offset: 0xFF0

Reset value: 0x0071 3111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG6[3:0]				CFG5[3:0]			
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **CFG6[3:0]**: Hardware configuration 6
This bitfield returns the generic value DEFAULT_DAPCID.

Bits 19:16 **CFG5[3:0]**: Hardware configuration 5
This bitfield returns the generic value DEFAULT_TDCID.

Bits 15:12 **CFG4[3:0]**: Hardware configuration 4
This bitfield returns the generic value CID_WIDTH.

Bits 11:8 **CFG3[3:0]**: Hardware configuration 3
This bitfield returns the generic value PRIV_ENABLED.

Bits 7:4 **CFG2[3:0]**: Hardware configuration 2
This bitfield returns the generic value SEC_ENABLED.

Bits 3:0 **CFG1[3:0]**: Hardware configuration 1
This bitfield returns the generic value RIF_ENABLED.

8.5.14 RIFSC version register (RIFSC_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: RIFSC major revision

Bits 3:0 **MINREV[3:0]**: RIFSC minor revision

8.5.15 RIFSC identification register (RIFSC_IPIDR)

Address offset: 0xFF8

Reset value: 0x0010 0162

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RIFSC identification code

8.5.16 RIFSC size identification register (RIFSC_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RIFSC size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x04 (4-Kbyte address decoding)

8.5.17 RIFSC register map

Table 43. RIFSC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RIFSC_RISC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x010 + 0x004 * x (x = 0 to 5) Last address: 0x024	RIFSC_RISC_SECCFGRx	SEC{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028-0x02C	Reserved	Reserved																															

Table 43. RIFSC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030 + 0x004 * x (x = 0 to 5) Last address: 0x044	RIFSC_RISC_PRIVCFGRx	PRIV{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048-0x04C	Reserved	Reserved																															
0x050 + 0x004 * x (x = 0 to 5) Last address: 0x064	RIFSC_RISC_RCFGLOCKRx	RLOCK{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068-0x06C	Reserved	Reserved																															
0x100 + 0x008 * y (y = 0 to 127) Last address: 0x4F8	RIFSC_RISC_PERy_CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
	Reset value										0	0	0	0	0	0	0	0									0	0	0			0	0
0x104 + 0x008 * y (y = 0 to 127) Last address: 0x4FC	RIFSC_RISC_PERy_SEMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMID [2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x500-0xBFC	Reserved	Reserved																															
0xC00	RIFSC_RIMC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDEN	Res.	Res.	Res.	Res.	DAPCID [2:0]			Res.	TDCID [2:0]			Res.	Res.	WUCDONE	GLOCK
	Reset value																	1					1	1	1		0	0	1			0	0
0xC04	RIFSC_RIMC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCEN	GPDEN
	Reset value																															0	0
0xC08-0xC0C	Reserved	Reserved																															
0xC10 + 0x004 * x (x = 1 to 15) Last address: 0xC4C	RIFSC_RIMC_ATTRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPRIV	MSEC	Res.	MCID[2:0]			Res.	CIDSEL	Res.	Res.
	Reset value																							0	0		0	0	0		0		
0xC50 to 0xFAC	Reserved	Reserved																															
0xFB0 + 0x004 * x (x = 0 to 3) Last address: 0xFBC	RIFSC_PPSRx	PPEN{i + 32 * x}																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFC0-0xFE4	Reserved	Reserved																															
0xFE8	RIFSC_HWCFGR3	CFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFEC	RIFSC_HWCFGR2	CFG3[7:0]								CFG2[7:0]								CFG1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 43. RIFSC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	RIFSC_ HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG6[3:0]				CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value									0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1
0xFF4	RIFSC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	RIFSC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0
0xFFC	RIFSC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

1. See *RIFSC peripheral protection status register x (RIFSC_PPSRx)* for the various reset values.

Refer to *Section 2.4* for the register boundary addresses.

9 Resource isolation slave unit for address space protection (block-based) (RISAB)

9.1 RISAB introduction

The RIF (resource isolation framework) is a comprehensive set of hardware blocks designed to enforce and manage isolation of STM32 hardware resources, such as memory and peripherals.

Through RISAB registers, a trusted domain application, or the application to whom the page configuration has been delegated assigns memory pages to one or more security domains (secure, privilege, compartment).

9.2 RISAB main features

- As an RIF core component, the RISAB assigns specific device resources to security domains:
  - Isolating resources defined as fixed sized embedded memory blocks and pages
- Isolation of hardware memory resources based on multiple criteria:
  - Restricted to one or more hardware compartments
  - Secure-only or nonsecure-only
  - Privileged-only or privileged/unprivileged
  - Read-only, write-only, or read/write
- The trusted domain can delegate the resource isolation configuration to the processing environment that owns a page resource.
- Support to RIF special debug domain, with greater accesses for debugger used in secure or non-secure global debug sessions.

9.3 RISAB implementation

Table 44 describes the characteristics of the available instances.

Table 44. RISAB resource assignment

Protected memory	Resource name(1)	Blocks per pages(2)	Number of pages	Memory size (Kbytes)	Default protection(3)
SYSRAM	RISAB1(4)	8	32	128	Secure, privileged or unprivileged, any CID
	RISAB2(5)				
RETRAM	RISAB5				
SRAM1	RISAB3		16	64	Non-secure, privileged or unprivileged, any CID

1. According to the resource name, the reset values of RISAB_HWCFGR1 change, they are 0x05393111 for RISAB1/2/5 and 0x04393111 for RISAB3. Similarly, the reset values of RISAB_HWCFGR2 change: 0x00020000 for RISAB1/2/5, 0x00010000 for RISAB3.
2. The block size is fixed to 512 bytes, the page size to 4 Kbytes.
3. Default protection policy out of reset. When default protection is secure, the reset value of RISAB_PGy_SECCFGR is 0x0000 00FF (instead of 0x0000 0000).

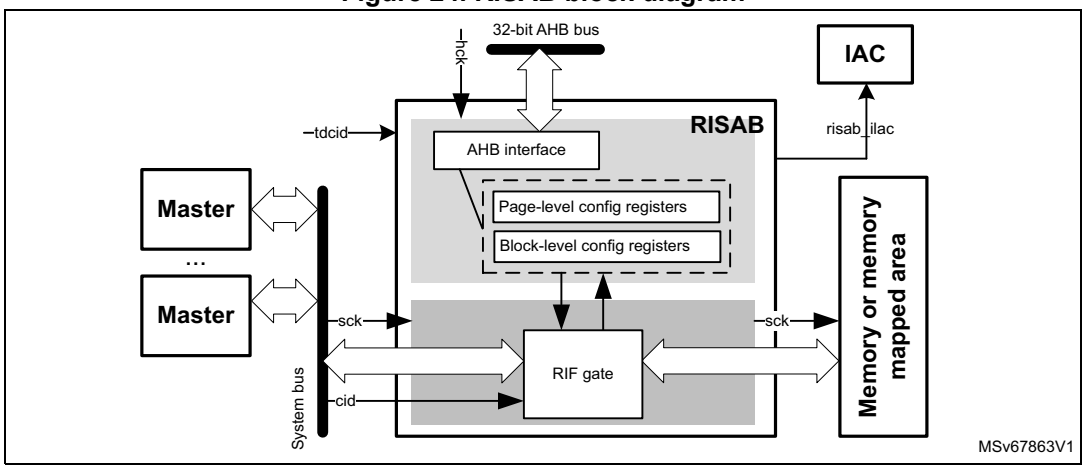
4. Manages pages 0 to 31 of the 256-Kbyte SYSRAM. Only Cortex-A35 can program RISAB1.
5. Manages pages 32 to 63 of the 256-Kbyte SYSRAM. Only Cortex-A35 can program RISAB2.

9.4 RISAB functional description

9.4.1 RISAB block diagram

Figure 24 shows the block diagram of the RISAB, associated with its memory controller on the interconnect (optional), and the IAC peripheral.

Figure 24. RISAB block diagram



9.4.2 RISAB internal signals

Table 45 describes the signals available at RISAB level, not at product level (on pads).

Table 45. RISAB internal input/output signals

Name	Type	Description
risab_sck	Digital input	System bus clock, also used by the protected memory controller
risab_hclk		AHB bus clock
risab_ilac	Digital output	Illegal access signal to IAC

9.4.3 RISAB reset and clocks

The RISAB peripheral has two clocks: one for its AHB register interface, and one for its system bus interface. The system bus clock is shared with the system bus memory controller target.

The RISAB peripheral has a single reset input.

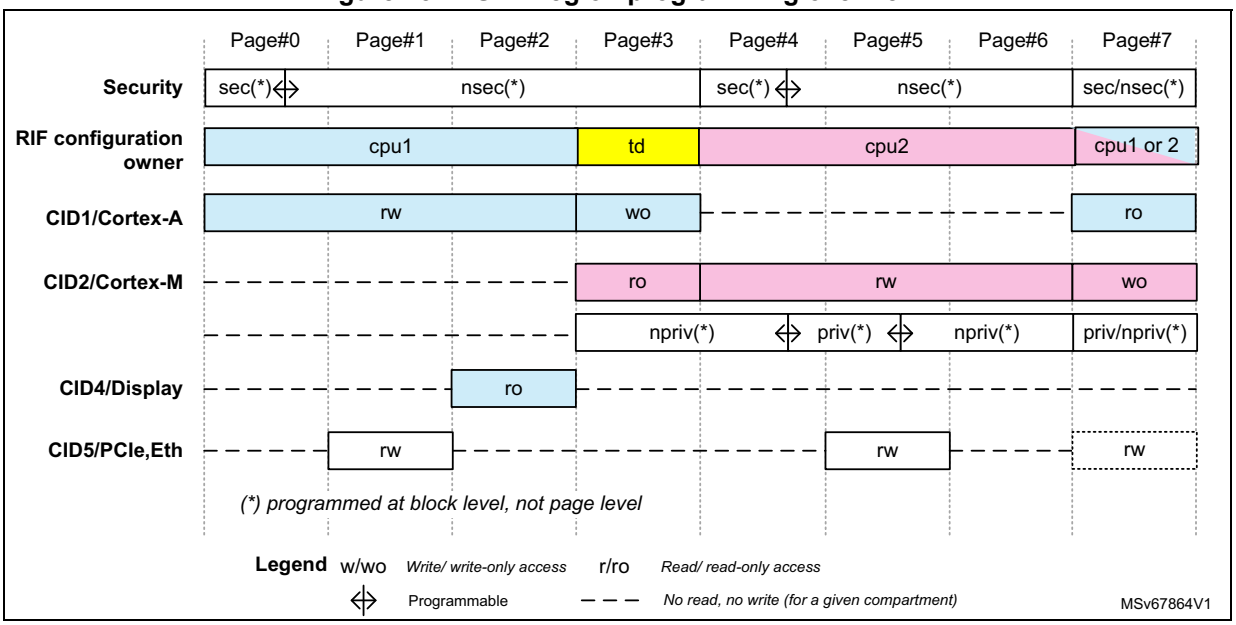
Default RISAB values after a device general reset (power-on reset) are defined in *Section 9.3*.

9.4.4 RISAB address space management

Through RISAB registers, a trusted domain application, or the application to which the page configuration has been delegated, assigns memory pages to one or more security domains (secure, privilege, compartment). *Figure 25* illustrates the principle of this allocation.

Note: Page size is defined in Section 9.3.

Figure 25. RISAB region programming overview



As a Cortex-M defines the configurable regions in its address space with a 256-byte granularity, the RISAB can also be used to allocate secure and privileged access protection, at a lower 512-byte block level. The application can then align Cortex-M, and RISAB secure and privileged regions allocation.

At any time, the trusted domain application can lock the RIF configuration of a page until the next reset, by setting the RLOCK bit in RISAB_RCFGLOCKR.

As defined in *Section 5.4: Resource isolation framework (RIF)*, a debug domain (privileged) must be granted the following accesses, depending on its security:

- A secure debug domain can read and write any RISAB pages. It can also modify registers reserved to the trusted domain.
- For reads (writes), a non-secure debug domain can only access non-secure pages, readable (writeable) by at least one compartment.

Access rules to secure, non-secure, privileged, and unprivileged memory regions are described in *Section 5.4.2: RIF access control*.

9.4.5 RISAB programming

When an application needs to use page x there are two configuration options:

- The trusted domain programs the RIF configuration for this page. In this case, DCEN is cleared in RISAB_PGx_CIDCFGR.
- The trusted domain delegates the RIF configuration to a single CID, defined with DCCID in RISAB_PGx_CIDCFGR. In this case, DCEN must be set in this register.

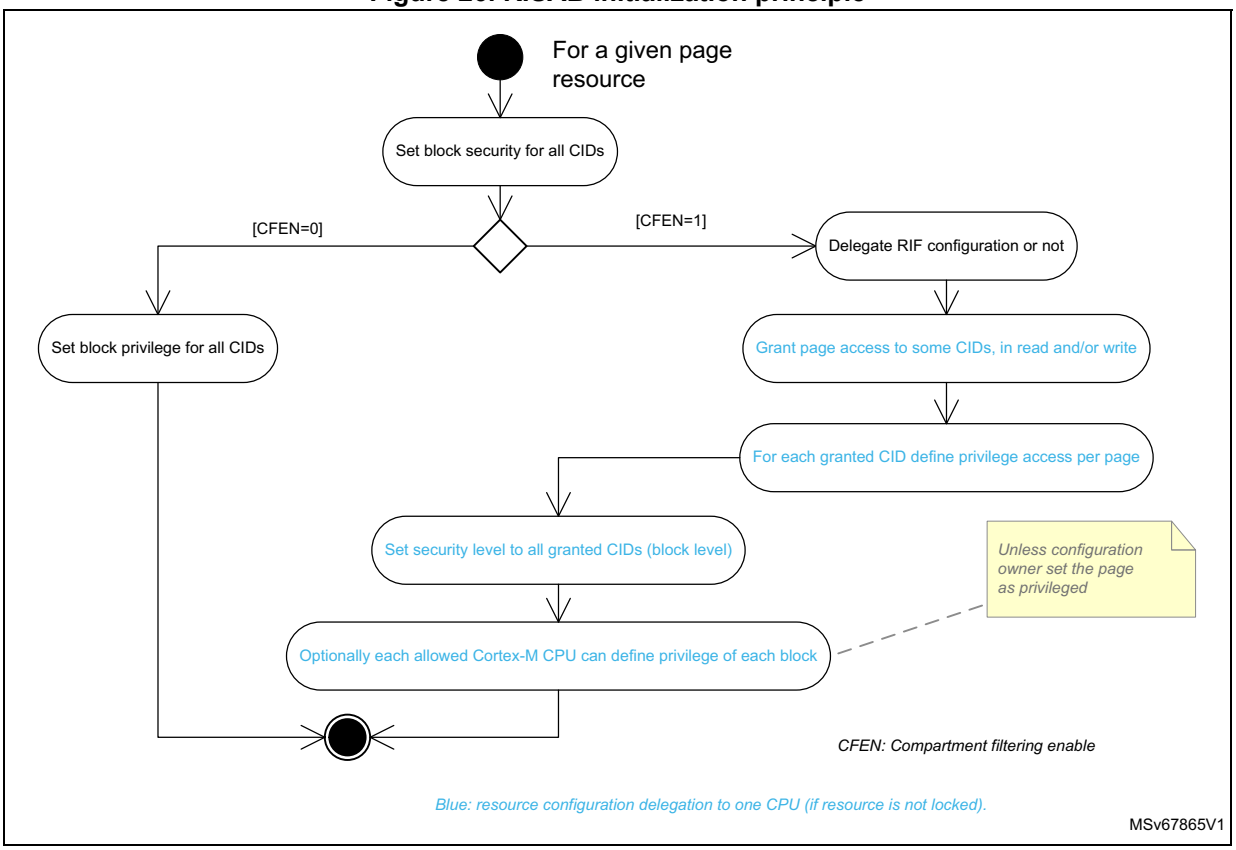
To configure a page x go through the following steps (block and page sizes are defined in *Section 9.3*):

1. Define the security level of each block in the page, using RISAB_PGx_SECCFGR.
2. Define which CIDs can access the page in read and write, and with which privilege, using, respectively RISAB_CIDyRDCFGR, RISAB_CIDyWRCFGR, and RISAB_CIDyPRIVCFGR.
3. Each Cortex-M CPUz can define, if applicable, the privileged level of each block in the page, using RISAB_PGx_CzPRIVCFGR, unless the whole page has been defined privileged by the configuration owner, using RISAB_CIDzPRIVCFGR.

Note: When at least one block is defined secure in RISAB_PGx_SECCFGR, the read, write, and privileged configuration at page level becomes secure-only. This is important when the RIF configuration is delegated (DCEN = 1).

The RISAB initialization method is summarized in *Figure 26*.

Figure 26. RISAB initialization principle



9.4.6 RISAB runtime modification of region or subregion configuration

The RISAB supports on-the-fly update of blocks and pages programming.

9.4.7 Managing illegal accesses

As a RIF component, the RISAB does not use the standard interrupt lines towards NVIC or GIC concentrators. The IAC captures all RIF interrupt events, called illegal access events.

Refer to *Section 11: Illegal access controller (IAC)* for more information on the way illegal accesses are managed.

9.5 RISAB registers

These registers are accessible through the register interface of each RISAB instance. Only word accesses (32 bits) are supported.

If the debug domain feature is activated and security and privileged policy allows it, the debug domain traffic goes through, regardless of any CID filtering policy. If the debug domain feature is deactivated, the CID filtering policy also applies to debug domain traffic.

9.5.1 RISAB configuration register (RISAB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRWIAD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bit 31 **SRWIAD**: Secure read/write illegal access disable

When set, this bit allows secure read/write data accesses to nonsecure blocks and pages (secure execute remains illegal).

0: Any secure access to nonsecure blocks or pages is illegal.

1: Secure read/write data access is allowed to nonsecure blocks or pages, while secure execute generates a bus error.

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Global lock

This bit locks the configuration of RISAB registers until next reset. This bit is cleared by default and, once set, cannot be reset until global RISAB reset.

0: RISAB registers are writable.

1: All writes to RISAB registers are ignored, except RISAB_IACR.

9.5.2 RISAB illegal access status register (RISAB_IASR)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged trusted domain CID read access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	CAEF
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IAEF**: Illegal access error flag

This bit is set when an illegal access is detected on the system bus. More details on the error can be found in RISAB_IAESR and RISAB_IADDR.

This bit is cleared by setting the corresponding bit in RISAB_IACR.

Bit 0 **CAEF**: configuration access error flag

This bit is set when an illegal access to any RISAB configuration register is detected. This bit is cleared by setting the corresponding bit in RISAB_IACR.

9.5.3 RISAB illegal access clear register (RISAB_IACR)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged trusted domain CID write access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	CAEF
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IAEF**: Illegal access error flag

Set this bit to clear IAEF in RISAB_IASR, allowing the capture of new error information in RISAB_IAESR and RISAB_IADDR.

Note: Clearing this bit does not clear RISAB_IADDR.

Bit 0 **CAEF**: Configuration access error flag

Set this bit to clear CAEF in RISAB_IASR.

9.5.4 RISAB configuration lock register (RISAB_RCFGLOCKR)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAB_CR. Refer to *Section 9.3* for the actual number of blocks per page.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK 31	RLOCK 30	RLOCK 29	RLOCK 28	RLOCK 27	RLOCK 26	RLOCK 25	RLOCK 24	RLOCK 23	RLOCK 22	RLOCK 21	RLOCK 20	RLOCK 19	RLOCK 18	RLOCK 17	RLOCK 16
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK 15	RLOCK 14	RLOCK 13	RLOCK 12	RLOCK 11	RLOCK 10	RLOCK 9	RLOCK 8	RLOCK 7	RLOCK 6	RLOCK 5	RLOCK 4	RLOCK 3	RLOCK 2	RLOCK 1	RLOCK 0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **RLOCKy**: Resource lock for page y (y = 31 to 0)

This bit is set to lock page y resource. It is cleared by default and, once set, cannot be cleared until RISAB is reset.

0: RISAB_PGy_CIDCFGR, RISAB_PGy_SECCFGR, RISAB_PGy_PRIVCFGR, and RISAB_PGy_C2PRIVCFGR are writable. Bit y in RISAB_CIDxPRIVCFGR, RISAB_CIDxRDCFGR, and RISAB_CIDxWRCFGR are writable (x = 0 to 6).

1: Writes to RISAB_PGy_CIDCFGR, RISAB_PGy_SECCFGR, RISAB_PGy_PRIVCFGR, and RISAB_PGy_C2PRIVCFGR are ignored. Writes to bit y in RISAB_CIDxPRIVCFGR, RISAB_CIDxRDCFGR, and RISAB_CIDxWRCFGR are ignored (m = 0 to 6).

9.5.5 RISAB illegal access error status register (RISAB_IAESR)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged trusted domain CID read access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IANRW	Res.	IASEC	IAPRIV	Res.	IACID[2:0]		
								r		r	r		r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **IANRW**: Illegal access read/write

When IAEF = 1 in RISAB_IASR, IANRW captures the access type of the illegal access that occurred on the system bus.

0: Illegal access was a data read.

1: Illegal access was a data write.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **IASEC**: Illegal access security

When IAEF = 1 in RISAB_IASR, IASEC captures the security type of the illegal access that occurred on the system bus.

0: Illegal access was nonsecure.

1: Illegal access was secure.

Bit 4 **IAPRIV**: Illegal access privileged
When IAEF = 1 in RISAB_IASR, IAPRIV captures the privilege state of the master that issued the illegal access on the system bus.
0: Illegal access was unprivileged.
1: Illegal access was privileged.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **IACID[2:0]**: illegal access compartment ID
When IAEF = 1 in RISAB_IASR, IACID[2:0] captures the compartment ID of the master that issued the illegal access on the system bus.

9.5.6 RISAB illegal address register (RISAB_IADDR)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged trusted domain CID read access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IADD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IADD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IADD[31:0]**: Illegal address
When IAEF = 1 in RISAB_IASR, IADD[31:0] captures the address of the erroneous access. Additional information can be found in RISAB_IAESR.

9.5.7 RISAB page y security configuration register (RISAB_PGy_SECCFGR)

Address offset: 0x100 + 0x4 * y (y = 0 to 31)

Reset value: 0x0000 00XX (refer to *Section 9.3* for the actual value)

Nonsecure or unprivileged writes to this register are ignored, all reads are allowed. Writes are ignored if GLOCK is set in RISAB_CR, or RLOCKy is set in RISAB_RCFGLOCKR. Refer to *Section 9.3* for the actual number of blocks per page.

When the compartment filtering and delegated configuration are enabled (CFEN = DCEN = 1 in RISAB_PGy_CIDCFGR), only the compartment DCCID defined in RISAB_PGy_CIDCFGR can write to this register. When the compartment filtering is enabled but delegation is disabled (CFEN = 1, DCEN = 0), only the trusted domain CID can program this register.

When the compartment filtering is disabled (CFEN = 0), any compartment can program this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SECz**: Security configuration for block z (z = 7 to 0)

0: Only nonsecure requests can access block z of page y. Secure data access can be granted if SRWIAD = 1 in RISAB_CR (secure instruction fetch triggers a bus error).

1: Only secure requests can access block z of page y.

9.5.8 RISAB page y default privileged configuration register (RISAB_PGy_PRIVCFGR)

Address offset: 0x200 + 0x4 * y (y = 0 to 31)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, while all reads are allowed. Writes are ignored if GLOCK is set in RISAB_CR, or RLOCKy is set in RISAB_RCFGLOCKR. Refer to *Section 9.3* for the actual number of blocks per page.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVz**: Privileged configuration for block z (z = 7 to 0)

This bit is ignored if the filtering per compartment is activated for page y (CFEN = 1 in RISAB_PGy_CIDCFGR).

0: Privileged and unprivileged access are granted to block z of page y.

1: Only privileged access is granted to block z of page y.

Note: Nonsecure write to this bit is ignored if SECz is set in RISAB_PGy_SECCFGR.

9.5.9 RISAB page y privileged configuration register for compartment 2 (RISAB_PGy_C2PRIVCFGR)

Address offset: 0x600 + 0x4 * y (y = 0 to 31)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, all reads are allowed. Writes are ignored if GLOCK is set in RISAB_CR, or RLOCKy is set in RISAB_RCFGLOCKR.

Only a compartment 2 master can write to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVz**: Privileged configuration for block z (z = 7 to 0)

This bit is ignored if the filtering per compartment is not activated for page y (CFEN = 0 in RISAB_PGy_CIDCFGR).

0: Compartment 2 privileged and unprivileged accesses are granted to block z of page y, if PRIVy is cleared in RISAB_CID2PRIVCFGR. If PRIVy is set, only privileged access is granted.

1: Only compartment 2 privileged access is granted to block z of page y.

Note: Nonsecure write to this bit is ignored if SECz is set in RISAB_PGy_SECCFGR.

9.5.10 RISAB compartment x privilege configuration register (RISAB_CIDxPRIVCFGR)

Address offset: 0x800 + 0x20 * x (x = 0 to 6)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, any read is allowed. Writes are ignored if GLOCK is set in RISAB_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPRIV 31	PPRIV 30	PPRIV 29	PPRIV 28	PPRIV 27	PPRIV 26	PPRIV 25	PPRIV 24	PPRIV 23	PPRIV 22	PPRIV 21	PPRIV 20	PPRIV 19	PPRIV 18	PPRIV 17	PPRIV 16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRIV 15	PPRIV 14	PPRIV 13	PPRIV 12	PPRIV 11	PPRIV 10	PPRIV 9	PPRIV 8	PPRIV 7	PPRIV 6	PPRIV 5	PPRIV 4	PPRIV 3	PPRIV 2	PPRIV 1	PPRIV 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPRIVy**: Page y privileged configuration (y = 31 to 0)

PPRIVy is ignored if the filtering per compartment is not activated for page y (CFEN = 0 in RISAB_PGy_CIDCFGR).

Non-secure write to this bit is ignored if any bit is set in RISAB_PGy_SECCFGR, or if DCEN = 0 while CFEN = 1 in RISAB_PGy_CIDCFGR.

Writes are ignored if RLOCKy is set in RISAB_RCFGLOCKR.

0: Privileged and unprivileged access for compartment x are granted to page y.

Note: Additional privilege access control is defined in RISAB_PGy_C2PRIVCFGR.

1: Compartment x unprivileged accesses to page y are blocked (read as 0, write is ignored).

Note: When the delegated configuration is enabled (DCEN = 1 in RISAB_PGy_CIDCFGR), only the compartment DCCID defined in RISAB_PGy_CIDCFGR can write to this bit. When the delegation is disabled (DCEN = 0), only the trusted domain CID can write to this bit.

When the compartment filtering is disabled (CFEN = 0 in RISAB_PGy_CIDCFGR), any compartment can program this bit.

9.5.11 RISAB compartment x read configuration register (RISAB_CIDxRDCFGR)

Address offset: 0x808 + 0x20 * x (x = 0 to 6)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, any read is allowed. Writes are ignored if GLOCK is set in RISAB_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRDEN 31	PRDEN 30	PRDEN 29	PRDEN 28	PRDEN 27	PRDEN 26	PRDEN 25	PRDEN 24	PRDEN 23	PRDEN 22	PRDEN 21	PRDEN 20	PRDEN 19	PRDEN 18	PRDEN 17	PRDEN 16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRDEN 15	PRDEN 14	PRDEN 13	PRDEN 12	PRDEN 11	PRDEN 10	PRDEN 9	PRDEN 8	PRDEN 7	PRDEN 6	PRDEN 5	PRDEN 4	PRDEN 3	PRDEN 2	PRDEN 1	PRDEN 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRDENy**: Page y read access enable (y = 31 to 0)

PRDENy is ignored if the filtering per compartment is not activated for page y (CFEN = 0 in RISAB_PGy_CIDCFGR).

Non-secure write to this bitfield is ignored if any bit is set in RISAB_PGy_SECCFGR, or if DCEN = 0 while CFEN = 1 in RISAB_PGy_CIDCFGR.

Writes are ignored if RLOCKy is set in RISAB_RCFGLOCKR.

0: Compartment x read as 0 when accessing page y.

1: Compartment x can read page y.

Note: When the delegated configuration is enable (DCEN = 1 in RISAB_PGy_CIDCFGR), only the compartment DCCID defined in RISAB_PGy_CIDCFGR can write to this bit. When the delegation is disabled (DCEN = 0), only the trusted domain CID can write to this bit.

When compartment filtering is disabled (CFEN = 0 in RISAB_PGy_CIDCFGR), any compartment can program this bit.

9.5.12 RISAB compartment x write configuration register (RISAB_CIDxWRCFGR)

Address offset: 0x810 + 0x20 * x (x = 0 to 6)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, any read is allowed. Writes are ignored if GLOCK is set in RISAB_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PWR EN31	PWR EN30	PWR EN29	PWR EN28	PWR EN27	PWR EN26	PWR EN25	PWR EN24	PWR EN23	PWR EN22	PWR EN21	PWR EN20	PWR EN19	PWR EN18	PWR EN17	PWR EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWRE N15	PWRE N14	PWRE N13	PWRE N12	PWRE N11	PWRE N10	PWR EN9	PWR EN8	PWR EN7	PWR EN6	PWR EN5	PWR EN4	PWR EN3	PWR EN2	PWR EN1	PWR EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PWRENy**: Page y write access enable (y = 31 to 0)

PWRENy is ignored if the filtering per compartment is not activated for page y (CFEN = 0 in RISAB_PGy_CIDCFGR).

Non-secure write to this bitfield is ignored if any bit is set in RISAB_PGy_SECCFGR, or if DCEN = 0 while CFEN = 1 in RISAB_PGy_CIDCFGR.

Writes are ignored if RLOCKy is set in RISAB_RCFGLOCKR.

0: Compartment x write to page y is ignored.

1: Compartment x can write page y.

Note: When the delegated configuration is enable (DCEN = 1 in RISAB_PGy_CIDCFGR), only the compartment DCCID defined in RISAB_PGy_CIDCFGR can write to this bit. When the delegation is disabled (DCEN = 0), only the trusted domain CID can write to this bit.
When compartment filtering is disabled (CFEN = 0 in RISAB_PGy_CIDCFGR), any compartment can program this bit.

9.5.13 RISAB page y CID configuration register (RISAB_PGy_CIDCFGR)

Address offset: 0xA00 + 0x4 * y (y = 0 to 31)

Reset value: 0x0000 0000

Secure privileged trusted domain CID write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAB_CR, or RLOCKx is set in RISAB_RCFGLOCKR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID[2:0]			Res.	DCEN	Res.	CFEN
									rw	rw	rw		rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **DCCID[2:0]**: Delegated configuration CID

The trusted domain application can use this bitfield to statically define which compartment can program the security domain of page y, if DCEN = 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCEN**: Delegated configuration enable

This bit defines the access control of RISAB_PGy_SECCFGR, PPRIVy in RISAB_CIDxPRIVCFGR, PRDENy in RISAB_CIDxRDCFGR, and PWRENy in RISAB_CIDxWRCFGR.

0: Bit/register can be written only by trusted domain application.

1: Bit/register can be written only by the compartment DCCID[2:0], defined in this register.

Note: When DCEN is set, the configuration application must be secure if at least one bit is set in RISAB_PGy_SECCFGR.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CFEN**: CID filtering enable

0: Compartment-based filtering is disabled for page y.

1: Compartment-based filtering is enabled for page y.

Note: For any CFEN value, the access controls to page y, defined in RISAB_PGy_SECCFGR, and RISAB_zzPRIVCFGR (zz = PGy_ if CFEN = 0, zz=PGy_C2 or CIDx if CFEN = 1) always apply.

9.5.14 RISAB hardware configuration register 3 (RISAB_HWCFGR3)

Address offset: 0xFE8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CFG[31:0]**: Hardware configuration

This bitfield returns 0x0.

9.5.15 RISAB hardware configuration register 2 (RISAB_HWCFGR2)

Address offset: 0xFEC

Reset value: 0x000X 0000

Refer to *Section 9.3* for the actual reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CFG[31:0]**: Hardware configuration

This bitfield returns the protected memory size in bytes.

9.5.16 RISAB hardware configuration register 1 (RISAB_HWCFGR1)

Address offset: 0xFF0

Reset value: 0x0X39 3111

Refer to *Section 9.3* for the actual reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CFG7[3:0]				CFG6[3:0]				CFG5[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CFG7[3:0]**: Hardware configuration 7
This bitfield returns the number of pages (LOG_NUM_PAGE).

Bits 23:20 **CFG6[3:0]**: Hardware configuration 6
This bitfield returns the number of blocks per pages (LOG_BLOCKS_PER_PAGE).

Bits 19:16 **CFG5[3:0]**: Hardware configuration 5
This bitfield returns the block size (LOG_BLOCK_SIZE).

Bits 15:12 **CFG4[3:0]**: Hardware configuration 4
This bitfield returns the generic value CID_WIDTH.

Bits 11:8 **CFG3[3:0]**: Hardware configuration 3
This bitfield returns the generic value PRIV_ENABLED.

Bits 7:4 **CFG2[3:0]**: Hardware configuration 2
This bitfield returns the generic value SEC_ENABLED.

Bits 3:0 **CFG1[3:0]**: Hardware configuration 1
This bitfield returns the generic value RIF_ENABLED.

9.5.17 RISAB version register (RISAB_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**:RISAB major revision

Bits 3:0 **MINREV[3:0]**: RISAB minor revision

9.5.18 RISAB identification register (RISAB_IPIDR)

Address offset: 0xFF8

Reset value: 0x0010 016X

Refer to *Section 9.3* for the actual reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RISAB identification code

9.5.19 RISAB size identification register (RISAB_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RISAB size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x04 (4-Kbyte address decoding)

9.5.20 RISAB register map

Table 46. RISAB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RISAB_CR	SRWIAD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0																															0
0x004	Reserved	Reserved																															
0x008	RISAB_IASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	CAEF
	Reset value																															0	0
0x00C	RISAB_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	CAEF
	Reset value																															0	0
0x010	RISAB_ RCFGLOCKR	RLOCK31	RLOCK30	RLOCK29	RLOCK28	RLOCK27	RLOCK26	RLOCK25	RLOCK24	RLOCK23	RLOCK22	RLOCK21	RLOCK20	RLOCK19	RLOCK18	RLOCK17	RLOCK16	RLOCK15	RLOCK14	RLOCK13	RLOCK12	RLOCK11	RLOCK10	RLOCK9	RLOCK8	RLOCK7	RLOCK6	RLOCK5	RLOCK4	RLOCK3	RLOCK2	RLOCK1	RLOCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x01C	Reserved	Reserved																															
0x020	RISAB_IAESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IANRW	Res.	IASEC	IAPRIV	Res.	IACID [2:0]		
	Reset value																									0		0	0		0	0	0
0x024	RISAB_IADDR	IADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0x0FC	Reserved	Reserved																															
0x100 + 0x004 * y (y = 0 to 31) Last address: 0x17C	RISAB_PGy_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									1	1	1	1	1	1	1	1
0x180 to 0x1FC	Reserved	Reserved																															
0x200 + 0x4 * y (y = 0 to 31) Last address: 0x27C	RISAB_PGy_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x280 to 0x5FC	Reserved	Reserved																															
0x600 + 0x4 * y (y = 0 to 31) Last address: 0x67C	RISAB_PGy_ C2PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x680 to 0x7FC	Reserved	Reserved																															
0x800 + 0x020 * x (x = 0 to 6) Last address: 0x8C0	RISAB_ CIDxPRIVCFG R	PPRIV31	PPRIV30	PPRIV29	PPRIV28	PPRIV27	PPRIV26	PPRIV25	PPRIV24	PPRIV23	PPRIV22	PPRIV21	PPRIV20	PPRIV19	PPRIV18	PPRIV17	PPRIV16	PPRIV15	PPRIV14	PPRIV13	PPRIV12	PPRIV11	PPRIV10	PPRIV9	PPRIV8	PPRIV7	PPRIV6	PPRIV5	PPRIV4	PPRIV3	PPRIV2	PPRIV1	PPRIV0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 46. RISAB register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x808 + 0x20 * x (x = 0 to 6) Last address: 0x8C8	RISAB_ CIDxRDCFGR	PRDEN31	PRDEN30	PRDEN29	PRDEN28	PRDEN27	PRDEN26	PRDEN25	PRDEN24	PRDEN23	PRDEN22	PRDEN21	PRDEN20	PRDEN19	PRDEN18	PRDEN17	PRDEN16	PRDEN15	PRDEN14	PRDEN13	PRDEN12	PRDEN11	PRDEN10	PRDEN9	PRDEN8	PRDEN7	PRDEN6	PRDEN5	PRDEN4	PRDEN3	PRDEN2	PRDEN1	PRDEN0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x810 + 0x20 * x (x = 1 to 6) Last address: 0x8D0	RISAB_ CIDxWRCFGR	PWREN31	PWREN30	PWREN29	PWREN28	PWREN27	PWREN26	PWREN25	PWREN24	PWREN23	PWREN22	PWREN21	PWREN20	PWREN19	PWREN18	PWREN17	PWREN16	PWREN15	PWREN14	PWREN13	PWREN12	PWREN11	PWREN10	PWREN9	PWREN8	PWREN7	PWREN6	PWREN5	PWREN4	PWREN3	PWREN2	PWREN1	PWREN0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8D4-0x9FC	Reserved	Reserved																															
0xA00 + 0x4 * y (y = 1 to 31) Last address: 0xA7C	RISAB_PGy_ CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID [2:0]			Res.	DCEN	Res.	CFEN
	Reset value																										0	0	0	0	0	0	0
0xA80-0xFE4	Reserved	Reserved																															
0xFE8	RISAB_ HWCFGR3	CGF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFEC	RISAB_ HWCFGR2	CGF[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFF0	RISAB_ HWCFGR1	Res.	Res.	Res.	Res.	CFG6[3:0]				CFG6[3:0]				CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value(1)					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFF4	RISAB_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																										0	0	0	1	0	0	0	0
0xFF8	RISAB_IPIDR	ID[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFFC	RISAB_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

1. Refer to *Section 9.3* for the actual value.

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

10 Resource isolation slave unit for address space protection (full version) (RISAF)

10.1 RISAF introduction

The RIF (resource isolation framework) is a comprehensive set of hardware blocks designed to enforce and manage isolation of STM32 hardware resources, like memory and peripherals.

Through RISAF registers, a trusted domain application or the application to whom the configuration has been delegated, assigns memory regions and subregions to one or more security domains (secure, privileged, compartment).

RISAF includes the DDR memory cipher engine (DDRMCE) feature.

10.2 RISAF main features

- Assign variable sized address space regions and subregions to security domains.
- Make memory regions:
  - Restricted to one or more than one hardware compartments
  - Secure-only or nonsecure-only
  - Privileged-only or privileged/unprivileged
  - Read-only, write-only, or read/write
- The trusted domain can delegate configuration and positioning of a subregion (within a region) to another security domain.
- The trusted domain can delegate configuration and positioning of a subregion (within a nonsecure region) to a nonsecure privileged software.
- Support of special debug domain with greater accesses in either secure or nonsecure
- When applicable, the DDRMCE can encrypt/decrypt on-the-fly the traffic going to/from the DDR-SDRAM. There are two options:
  - Fast block cipher:
    - > Best performance (128-bit AES ECB cipher)
    - > Uses random key automatically drawn from the RNG peripheral at each power on
  - Block cipher:
    - > Best security (128- or 256-bit AES ECB cipher)
    - > Associated with a side-channel protected key derivation function based on Keccak-400 algorithm published on https://keccak.team

10.3 RISAF implementation

The following table describes the characteristics of the available RISAFs, including watermark programming details (such as granularity and protected address space).

Table 47. RISAF resource assignment

Protected memory	Resource name(1)	Number of base regions	Subregions per base region	Granularity (bytes)	Address space size and alignment(2)	DDRMCE option	Dual port option	Default protection (3)
BKPSRAM	RISAF1	4	2	256	8 Kbytes	No	No	Primary region only: secure, privileged, TDCID only
OCTOSPI1	RISAF2	4		4096	256 Mbytes			
DDR	RISAF4	15			4 Gbytes	Yes	Yes	

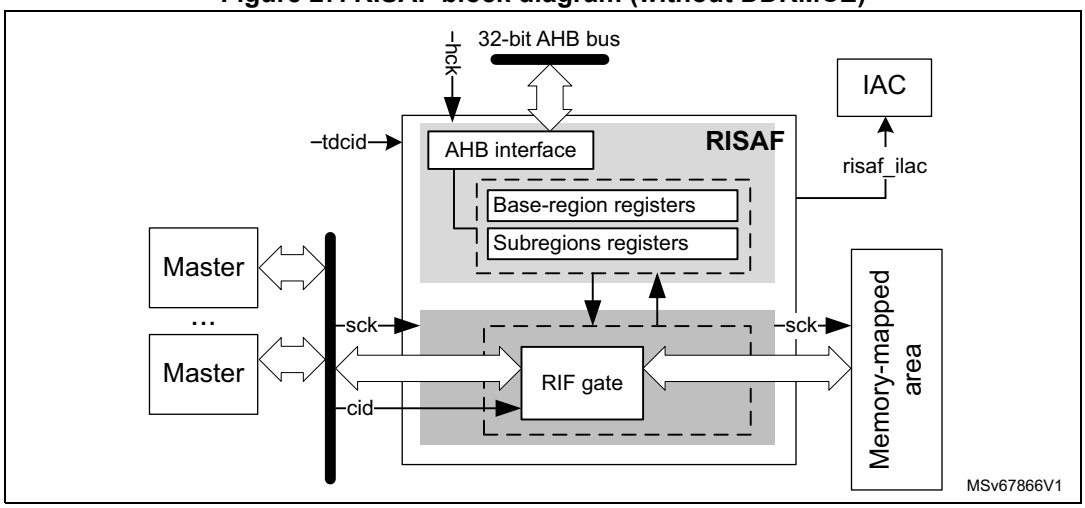
1. According to the resource name, the reset value of RISAF_HWCFGR changes: 0x05080505 for RISAF1, 0x100C0505 for RISAF2, and 0x140C1010 for RISAF4. The reset value of RISAF_IPIDR is 0x00100166 for RISAF1 and RISAF2, and 0x00100167 for RISAF4. The reset value of RISAF_VERR is 0x0000 0010 for RISAF1 and RISAF2, 0x0000 0020 for RISAF4. The reset value of RISAF_SIDR is 0xA3C5 DD01 for RISAF1 and RISAF2, and 0xA3C5 DD08 for RISAF4.
2. When address space size is the full 32-bit address space (4 Gbytes), watermarks are programmed using physical addresses, on 32 bits.
3. Default protection policy out of reset.

10.4 RISAF functional description

10.4.1 RISAF block diagram

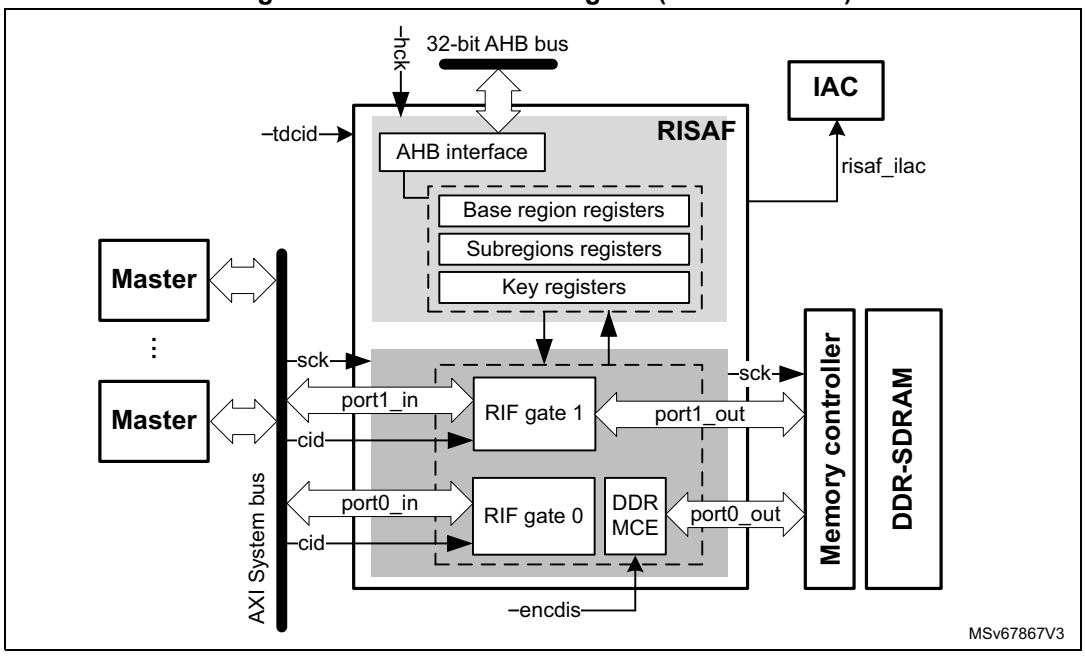
The following figures show the block diagrams of the two RISAF types (single port or dual port with encryption) associated with the IAC, and optionally, a memory controller. Both types are programmed the same way.

Figure 27. RISAF block diagram (without DDRMCE)



1. The tdcid input value is equal to TDCID.

Figure 28. RISAF4 block diagram (with DDRMCE)



10.4.2 RISAF internal signals

Table 48 describes internal signals available at RISAF level, not at product level (on pins).

Table 48. RISAF internal input/output signals

Name	Type	Description
risaf_sck	Digital input	System bus clock, also used by the protected memory controller
risaf_hclk		AHB bus clock
risaf_ilac	Digital output	Illegal access signal to IAC
risaf_encdis	Digital input	Static control signal disabling encryption

10.4.3 RISAF reset and clocks

The RISAF peripheral has two clocks, one for its AHB register interface, and one for its system bus interfaces. The system bus clock is shared with the system bus memory controller target.

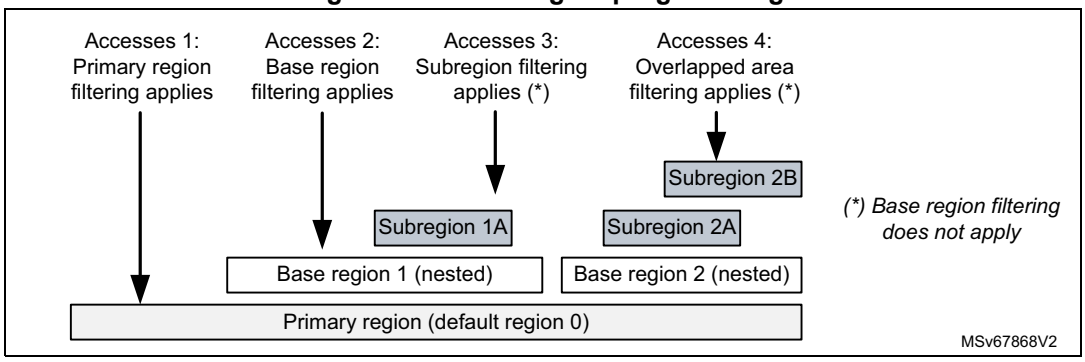
The RISAF peripheral has a single reset input.

Address spaces protected by the RISAF are by default secure, privileged, and TDCID only.

10.4.4 RISAF address space management

Through RISAF registers, a trusted domain application, or the application to whom the configuration has been delegated, assigns memory regions to one or more security domains (secure, privilege, compartment ID). *Figure 29* illustrates the principle of this allocation.

Figure 29. RISAF region programming



As shown in *Figure 29*, there are multiple types of access filtered by RISAF:

- Accesses 1 - Default region: access to an area not belonging to any enabled base region. Filtering for such access is fixed to secure, privileged and TDCID only.
- Accesses 2 - Application region > 0: access to an area in an enabled base region, not belonging to any enabled subregion. Filtering for such access is defined in the base region, where one or more compartments can be granted specific accesses.
- Accesses 3 - Non overlapped subregion: access to an area in an enabled nested subregion. Filtering for such access is defined in the subregion, with only one compartment that can be granted access.
- Accesses 4 - Overlapped subregion: accesses to an area overlapping two enabled nested subregions. In this case, filtering rules for these subregions are used by the RISAF as follows:
  - If both subregions are secure (resp. nonsecure), the overlap is secure (resp. nonsecure).
  - If a subregion is nonsecure (resp. unprivileged), the overlap is nonsecure (resp. unprivileged).
  - If a subregion grants access to a compartment, the access can go through. Up to two compartments can be granted access to an overlapped subregion.

Note: When a base or subregion is disabled, the associated filtering is disabled as well. For access types 3 or 4 (detailed above), base region filtering rules do not apply. When two enabled regions overlap, if one region grants access, the access goes through.

As defined in *Section 5.4: Resource isolation framework (RIF)*, a debug domain (privileged) must be granted the following accesses, depending on its security:

- The secure debug domain can read and write anywhere in the address space protected by RISAF. It can also modify the registers reserved to the trusted domain.
- For reads (resp. writes), the nonsecure debug domain can only access regions that are tagged by an application as nonsecure, readable (resp. writable) by at least one compartment.

Access rules to secure, nonsecure, privileged, and unprivileged memory regions are described in *Section 5.4: Resource isolation framework (RIF)*.

10.4.5 RISAF programming

Only the trusted domain can set up the access configuration to base region x through RISAF_REGx_CFGR, RISAF_REGx_CIDCFGR, RISAF_REGx_STARTR, and RISAF_REGx_ENDR.

The trusted domain, or the application to which the subregion z configuration has been delegated in RISAF_REGx_zNESTR, can set up the access configuration to subregion z of base region x through RISAF_REGx_zCFGR, RISAF_REGx_zSTARTR, and RISAF_REGx_zENDR (z = A or B).

As shown in *Table 49*, subregion right options depend upon the base region privileged and security rights.

Table 49. RISAF subregion security setup matrix

Base region rights	Nested subregion rights			
	Secure, privileged	Secure, unprivileged	Nonsecure, privileged	Nonsecure, unprivileged
Secure, privileged	OK(1)	OK	OK	OK
Secure, unprivileged	Not allowed	OK(1)	Not allowed	OK
Nonsecure, privileged	Not allowed	Not allowed	OK(1)	OK
Nonsecure, unprivileged	Not allowed	Not allowed	Not allowed	OK(1)

1. Default configuration

Configuring base regions in RISAF

The number of regions, the region watermark granularity, and the address space size depend upon the RISAF instance (see *Section 10.3*).

If the trusted domain must use a base region x (for example at boot time), the following steps are recommended.

1. The trusted domain sets start and end addresses for each of the required base region x, using RISAF_REGx_STARTR and RISAF_REGx_ENDR:
   - These registers are initialized with 32-bit physical addresses when the protected address space size is 4 Gbytes. For different sizes, refer to the register description.
   - Base regions must be defined within their primary region. The hardware enforces this rule if the application does not follow it.
2. For each of the above region x, the trusted domain programs RISAF_REGx_CIDCFGR, granting read and/or write access to any compartment.
3. For each of the above region x, the trusted domain finalizes the programming by configuring RISAF_REGx_CFGR, setting up security parameters (secure or encrypted if applicable), and granting privileged-only access, or not, to each compartment that has access to the region (see step *2*).
4. After a final check on the above configuration, the trusted domain enables each required base region x, setting the BREN bit in RISAF_REGx_CFGR.

At any time the trusted domain can lock the RISAF configuration until the next reset, by setting the GLOCK bit in RISAF_CR. Subregion registers are the exception to this rule, as explained in *Configuring subregions in RISAF*.

If the RISAF instance supports encryption (see *Section 10.3*), and if the trusted application requires region x to be encrypted, the recommended steps are:

- When ENC[1:0] = 0x2, the AES key is used directly without key derivation and is shared with all similarly encrypted regions, hence it is recommended to have all these regions secure.
- When ENC[1:0] = 0x1 side-channel protection is activated, impacting performance.
- When ENCDIS = 1 in RISAF_SR, or when the AXI port that receives the transaction does not support encryption (see port 1 in *Figure 28*), all writes to this base region x are ignored and all read requests return 0 if ENC[1:0] = 0x1 or 0x2 and BREN is set in RISAF_REGx_CFGR.

Configuring subregions in RISAF

Note: The number of subregions, the subregion watermark granularity, and the primary region address space size depend on the RISAF resource described in Section 10.3.

When an application needs to use the subregion z defined in the base region x, there are two configuration options in RISAF_REGx_zNESTR:

- The trusted domain programs the RIF configuration for this subregion. In this case DCEN is cleared.
- The secure privileged software delegates the RIF configuration to a single CID, defined with DCCID. In this case DCEN must also be set.

Configuring a subregion z consists in the following steps.

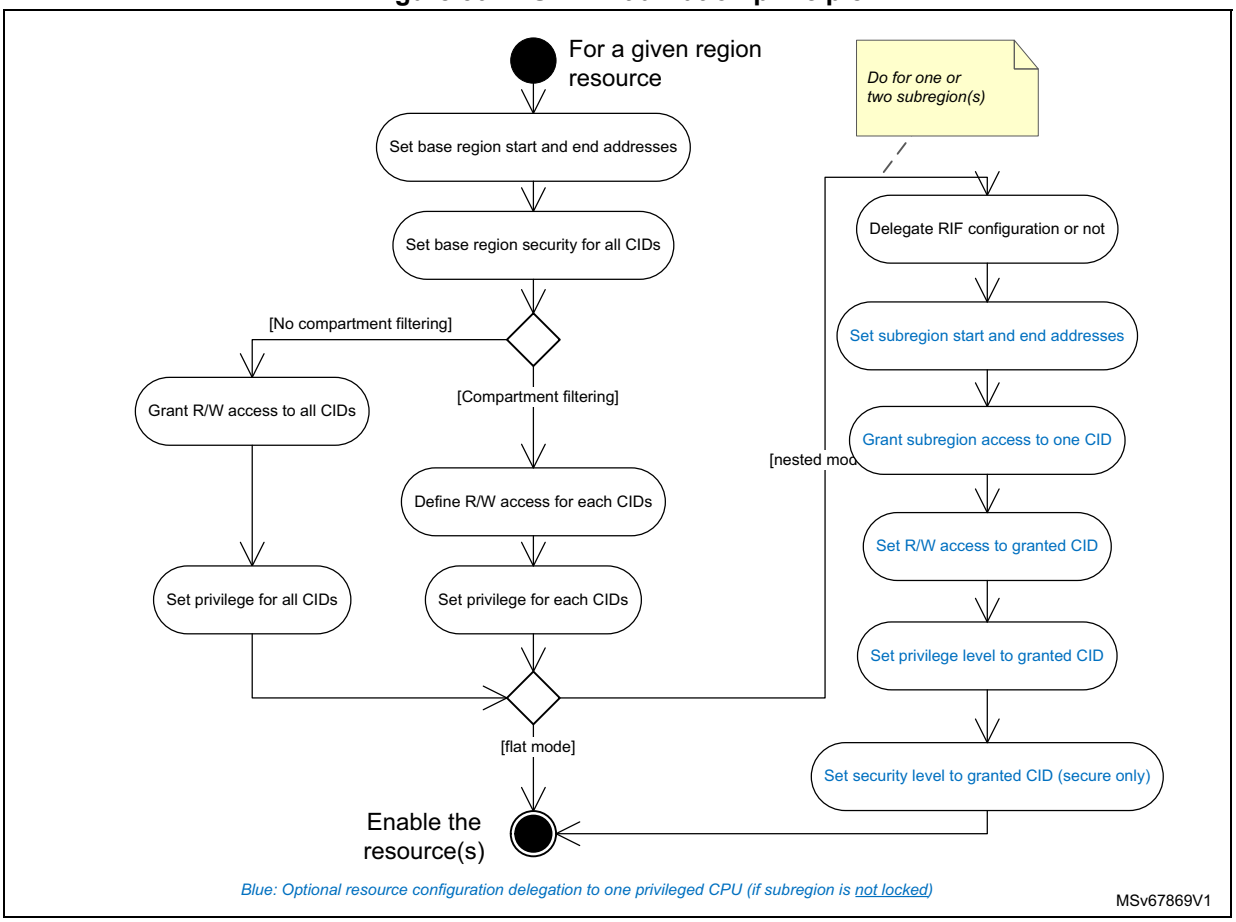
1. Define the start and end address of the subregion using RISAF_REGx_zSTARTR and RISAF_REGx_zENDR registers.
   - When programming subregion watermarks, use similar rules as for base regions (see *Configuring base regions in RISAF*).
   - Subregions must be defined within their base region. The RISAF enforces this rule if the application does not follow it.
2. Modify WREN and RDEN in RISAF_REGx_zCFGR to modify the access right of allowed compartment (defined with SCID bits).
3. In the same registers, modify the security and privileged access rights for the allowed compartment, if needed (see *Table 49* for applicable restrictions).
4. After a final check on the above configuration, the subregion must be enabled by setting SREN bit in RISAF_REGx_zCFGR.

After a trusted domain has locked all base regions (using GLOCK bit), subregion owners can lock their configuration using the RLOCK bit in RISAF_REGx_zCFGR.

Note: Only a secure application can change SEC in RISAF_REGx_zCFGR. If the base region is secure, only a secure application can configure subregion registers.

The overall RISAF initialization is summarized in *Figure 30*.

Figure 30. RISAF initialization principle



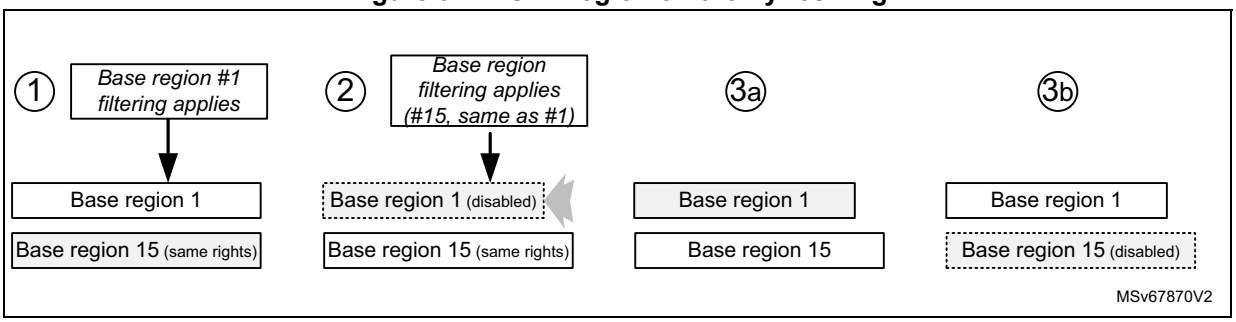
10.4.6 RISAF runtime modification of region or subregion configuration

The RISAF supports the on-the-fly update of the base region and subregion programming, except for the start- and end-address boundaries:

- To change RISAF_REGx_STARTR and RISAF_REGx_ENDR, the base region must be disabled (BREN = 0). In this case, the default primary region access rules apply immediately (secure, privileged, TDCID only).
- To change RISAF_REGx_zSTARTR and RISAF_REGx_zENDR (z = A or B), the subregion must be disabled (SREN = 0). In this case, the base region access rules apply immediately.

If the first solution introduces too many constraints, a workaround is shown in *Figure 31*.

Figure 31. RISAF region on-the-fly resizing



The recommended on-the-fly resizing of base region 1 becomes:

1. Define a spare base region (for example 15), exactly as current region 1.
2. Disable region 1 by clearing BREN.The region 15 access control (identical to region 1) immediately applies. No effect is experienced at system level. The region 1 can be safely resized.
3. Enable region 1, then disable region 15. The new region 1 access control immediately applies. Optionally, subregions can be added on top.

10.4.7 Managing illegal accesses

As a RIF component, the RISAF does not use the standard interrupt lines toward NVIC or GIC concentrators. The IAC captures all RIF interrupt events, called illegal access events.

Refer to *Section 11: Illegal access controller (IAC)* for more information on how illegal accesses are managed.

10.5 RISAF registers

The registers are accessible through the interface of the RISAF. Only word (32-bit) accesses are supported.

If the debug domain feature is activated and security and privileged policy allows it, the debug domain traffic goes through, regardless of any CID filtering policy. If the debug domain feature is de-activated the CID filtering policy also applies to debug domain traffic.

10.5.1 RISAF configuration register (RISAF_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed, writes are ignored if GLOCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Global lock

This bit locks the. configuration of RISAF registers until next reset. It is cleared by default, and, once set, cannot be reset until global RISAF reset.

0: RISAF registers are writable.

1: All writes to RISAF registers are ignored, except RISAF_IACR, RISAF_REGx_zCFGR, RISAF_REGx_zSTARTR, and RISAF_REGx_zENDR (z = A to B).

10.5.2 RISAF status register (RISAF_SR)

Address offset: 0x004

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC DIS	KEY RDY	KEY VALID
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ENCDIS**: Encryption disabled

This bit is reserved if the encryption feature is not available in RISAF (see *Section 10.3*). It is a non-volatile product configuration that cannot be changed by software.

0: When ENC[1:0] = 0x1 or 0x2 and BREN is set in RISAF_REGx_CFGR, all allowed write accesses in region x are encrypted, and all allowed read requests are decrypted.

1: When ENC[1:0] = 0x1 or 0x2 and BREN is set in RISAF_REGx_CFGR, all write accesses in region x are ignored, and all read requests return 0. In other words, the AES feature of RISAF is unusable, for any region.

Note: If BREN = 0, the AES feature of RISAF is not usable for a given region.

Bit 1 **KEYRDY**: Key ready

This bit is reserved if the encryption feature is not available in RISAF (see *Section 10.3*).

0: The most recently written key in RISAF_KEYRy is being expanded

1: The most recently written key in RISAF_KEYRy is ready for use

Bit 0 **KEYVALID**: Key valid

This bit is reserved if the encryption feature is not available in RISAF (see *Section 10.3*).

0: A valid key has not been written in RISAF_KEYRy

1: A valid key has been written in RISAF_KEYRy

10.5.3 RISAF illegal access status register (RISAF_IASR)

Address offset: 0x008

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF1	IAEF0	CAEF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IAEF1**: Illegal access error flag 1

This bit is reserved if the dual port feature is not available in the RISAF instance (see *Section 10.3*). The bit is set when an illegal access is detected by checker unit 1 on the system bus. More details on the error can be found in RISAF_IAESR1 and RISAF_IADDR1.

This bit is cleared by setting the corresponding bit in RISAF_IACR.

Bit 1 **IAEF0**: Illegal access error flag 0

This bit is set when an illegal access is detected by checker unit 0 on the system bus. More details on the error can be found in RISAF_IAESR0 and RISAF_IADDR0.

This bit is cleared by setting corresponding bit in RISAF_IACR.

Bit 0 **CAEF**: Configuration access error flag

This bit is set when an illegal access to any RISAF configuration register is detected. It is cleared by setting the corresponding bit in RISAF_IACR.

10.5.4 RISAF illegal access clear register (RISAF_IACR)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF1	IAEF0	CAEF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IAEF1**: Illegal access error flag 1

This bit is reserved if the dual port feature is not available in the RISAF instance (see *Section 10.3*). Set this bit to clear IAEF1 in RISAF_IASR, allowing the capture of a new error information from RIF gate 1 in RISAF_IAESR1 and RISAF_IADDR1.

Note: Clearing this bit does not clear RISAF_IADDR1.

Bit 1 **IAEF0**: Illegal access error flag 0

Set this bit to clear IAEF0 in RISAF_IASR, allowing the capture of a new error information from RIF gate 0 in RISAF_IAESR0 and RISAF_IADDR0 registers.

Note: Clearing this bit does not clear RISAF_IADDR0.

Bit 0 **CAEF**: Configuration access error flag

Set this bit to clear CAEF in RISAF_IASR.

10.5.5 RISAF illegal access error status register x (RISAF_IAESRx)

Address offset: 0x020 + 0x8 * x (x = 0 to 1)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only.

Note: RISAF_IAESR1 is reserved if the dual port feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IANRW	Res.	IASEC	IAPRIV	Res.	IACID[2:0]		
								r		r	r		r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **IANRW**: Illegal access read/write
When IAEFx = 1 in RISAF_IASR, IANRW captures the access type of the illegal access that has been detected by RIF gate x.
0: Illegal access was a data read or an instruction fetch
1: Illegal access was a data write

Bit 6 Reserved, must be kept at reset value.

Bit 5 **IASEC**: Illegal access security
When IAEFx = 1 in RISAF_IASR, IASEC captures the security type of the illegal access that has been detected by RIF gate x.
0: Illegal access was nonsecure
1: Illegal access was secure

Bit 4 **IAPRIV**: Illegal access privileged
When IAEFx = 1 in RISAF_IASR, IAPRIV captures the privilege state of the master that issued the illegal access detected by RIF gate x.
0: Illegal access was unprivileged
1: Illegal access was privileged

Bit 3 Reserved, must be kept at reset value.

Bits **2**:0 **IACID[2:0]**: Illegal access compartment ID
When IAEFx = 1 in RISAF_IASR, IACID[2:0] captures the compartment ID of the master that issued the illegal access detected by RIF gate x.

10.5.6 RISAF illegal address register x (RISAF_IADDRx)

Address offset: 0x024 + 0x8 * x (x = 0 to 1)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only.

Note: RISAF_IADDR1 is reserved if the dual port feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IADD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IADD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IADD[31:0]**: Illegal address
When IAEFx = 1 in RISAF_IASR, IADD[31:0] captures the address of the erroneous access through RIF gate x. Additional information can be found in RISAF_IAESRx. Captured address information is the byte offset from the base of the protected address space.

10.5.7 RISAF encryption key y (RISAF_KEYRy)

Address offset: 0x030 + 0x4 * y (y = 0 to 3)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Writes are ignored if GLOCK is set in RISAF_CR.

Note: RISAF_KEYR0/1/2/3 are reserved if the DDRMCE feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY{i + 32 * y}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY{i + 32 * y}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY{i + 32 * y}**: Cryptographic key (i = 31 to 0)

Write any value to this register to initialize this part of the key used by the DDRMCE when ENC[1:0]= 0x2.

The four key registers must be written in ascending order (KEYR0 to KEYR3) to set the KEYVALID bit. If this bit is not set after successful writing to those four registers, BSEC peripheral must be reset.

10.5.8 RISAF region x configuration register (RISAF_REGx_CFGR)

Address offset: 0x040 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 10.3* to verify the number of base regions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVC7	PRIVC6	PRIVC5	PRIVC4	PRIVC3	PRIVC2	PRIVC1	PRIVC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC[1:0]		Res.	Res.	Res.	Res.	Res.	SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
rw	rw						rw								rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **PRIVCy**: Privileged access for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1.

0: Application running in compartment y can access to region x in privileged and unprivileged mode.

1: Application running in compartment y can access to region x in privileged mode only.

Bits 15:14 **ENC[1:0]**: Encrypted region

This bitfield is taken into account only if BREN = 1 and if the DDRMCE feature is available in RISAF (see ENCDIS in RISAF_SR and *Section 10.3*).

0x0: No effect

0x1: Block cipher: all allowed write accesses are encrypted, all allowed read requests are decrypted. In this mode special side-channel protection is activated, impacting performance.

0x2: Fast block cipher: all allowed write accesses are encrypted, all allowed read requests are decrypted, when KEYRDY is set in RISAF_SR (to be used for secure regions only).

0x3: No effect

As the AES key is computed using the physical address information, ENC[1:0] = 0x1 must not be used only for secure regions. Key registers RISAF_KEYRy must be properly initialized before writing 0x2 to ENC[1:0] (KEYVALID set in RISAF_SR). Similarly, key registers RISAF_MKEYRy must be properly initialized before writing 0x1 to ENC[1:0] (MKVALID set in RISAF_XSR).

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **SEC**: Secure region

This bit is taken into account only if BREN = 1.

0: Only nonsecure requests can access region x. SEC value in RISAF_REGx_zCFGR is internally treated as 0-ed by RISAF (only nonsecure requests to subregions).

1: Only secure requests can access region x, and optional subregions can be defined as secure or nonsecure.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BREN**: Base region enable

0: Base region x is disabled. Access control of primary region 0 (secure, privileged, trusted domain TDCID) applies to any access between region x start and end addresses. Associated subregions are also disabled.

1: Base region x is enabled. Access controls defined in region x apply to any access between region x start and end addresses.

Note: The filtering starts from start address RISAF_REGx_STARTR (included) and ends at end address RISAF_REGx_ENDR (included).

10.5.9 RISAF region x start-address register (RISAF_REGx_STARTR)

Address offset: 0x044 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR, or if BREN is set in RISAF_REGx_CFGR.

Refer to *Section 10.3* to verify the number of base regions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BADDSTART[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADDSTART[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BADDSTART[31:0]**: Base region address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written to, and remain 0.

10.5.10 RISAF region x end-address register (RISAF_REGx_ENDR)

Address offset: 0x048 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

The reset value given here applies when the granularity is 4 Kbytes. If the granularity is 2^n, the reset value is 2^n - 1.

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR, or if BREN is set in RISAF_REGx_CFGR.

Refer to *Section 10.3* to verify the number of base regions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BADDEND[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADDEND[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BADDEND[31:0]**: Base region address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address in base region x.

Address bits less than the granularity cannot be modified, and remain 1.

Address bits greater than the protected address space size cannot be written, and remain 0.

10.5.11 RISAF region x CID configuration register (RISAF_REGx_CIDCFGR)

Address offset: 0x04C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 10.3* to verify the number of base regions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRENC7	WRENC6	WRENC5	WRENC4	WRENC3	WRENC2	WRENC1	WRENC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDENC7	RDENC6	RDENC5	RDENC4	RDENC3	RDENC2	RDENC1	RDENC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRENCy**: Write enable for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1 in RISAF_REGx_CFGR.

0: Application running in compartment y cannot write to region x (write ignored).

1: Application running in compartment y can write to region x. Additional control defined by SEC and PRIVCy in RISAF_REGx_CFGR, also applies.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RDENCy**: Read enable for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1 in RISAF_REGx_CFGR.

0: Application running in compartment y cannot read region x (read as 0).

1: Application running in compartment y can read region x. Additional control defined by SEC and PRIVCy in RISAF_REGx_CFGR, also applies.

10.5.12 RISAF region x subregion A configuration register (RISAF_REGx_ACFGR)

Address offset: 0x050 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 in this register.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register.

When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC[1:0]		WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID[2:0]			Res.	Res.	RLOCK	SREN
r	r	rw	rw			rw	rw		rw	rw	rw			rs	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ENC[1:0]**: Encrypted subregion

This read-only bitfield is a mirror to ENC[1:0] in RISAF_REGx_CFGR. It is reserved if the encryption feature is not available in RISAF (see ENCDIS in RISAF_SR, and *Section 10.3*).

0x0: No effect

0x1, 0x2: Allowed read (write) requests to subregion A of region x are decrypted (encrypted)

0x3: No effect

Bit 13 **WREN**: Write enable

This bit is taken into account only if SREN = 1 in this register.

0: Any write to subregion A of region x is ignored

1: Subregion A of region x can be written

Bit 12 **RDEN**: Read enable

This bit is taken into account only if SREN = 1 in this register.

0: Any read to subregion A of region x returns 0

1: Subregion A of region x can be read

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged subregion

This bit is taken into account only if SREN = 1 in this register.

0: Privileged and unprivileged accesses are granted in subregion A of region x

1: Only privileged accesses are granted in subregion A of region x

Note: When base region x is defined as unprivileged for the SCID compartment (PRIVCy = 0 in RISAF_REGx_CFGR, with SRCID = Cy), unprivileged accesses to subregion A are always granted.

Bit 8 **SEC**: Secure subregion

This bit is taken into account only if SREN = 1 in this register. Nonsecure modification to it are discarded.

0: Only nonsecure requests can access subregion A of region x

1: Only secure requests can access subregion A of region x

Note: When base region x is defined as nonsecure (SEC = 0 in RISAF_REGx_CFGR), only nonsecure accesses to subregion A are granted.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SRCID[2:0]**: Subregion CID

When SREN = 1 in this register, SRCID[2:0] defines which compartment has access to subregion A of RISAF region x.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RLOCK**: Resource lock

When this bit is set, the subregion resource is locked.

0: RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR are writable

1: Writes to RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR are ignored

Note: This bit is set once, cleared only by RISAF reset. It cannot be set if GLOCK = 0 in RISAF_CR.

Bit 0 **SREN**: Subregion enable

0: Subregion A is disabled. Access control of base region x applies to any access between this subregion start and end addresses.

1: Subregion A of region x is enabled. Access control defined in subregion A applies to any access between subregion A start and end addresses.

Note: The filtering starts from address RISAF_REGx_ASTARTR (included), and ends at address RISAF_REGx_AENDR (included).

10.5.13 RISAF region x subregion A start-address register (RISAF_REGx_ASTARTR)

Address offset: 0x054 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_ACFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register. When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADDSTART[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDSTART[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SADDSTART[31:0]**: Subregion address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in subregion A in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDSTART below the associated BADDSTART).

10.5.14 RISAF region x subregion A end-address register (RISAF_REGx_AENDR)

Address offset: 0x058 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

This reset value applies when the granularity is 4 Kbytes. More in general, if the granularity is 2^n, the reset value is 2^n - 1.

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_ACFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register. When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADDEND[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDEND[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SADDEND[31:0]**: Subregion address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address of subregion A in base region x.
Address bits less than the granularity cannot be modified, and remain 1.
Address bits greater than the protected address space size cannot be written, and remain 0.
The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDEND higher than the associated BADDEND).

10.5.15 RISAF region x subregion A nested mode register (RISAF_REGx_ANESTR)

Address offset: 0x05C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID[2:0]			Res.	DCEN	Res.	Res.
									rw	rw	rw		rw		

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **DCCID[2:0]**: Delegated configuration CID

The trusted domain can use this bitfield to statically define which compartment can program the RIF configuration of subregion A if DCEN = 1 in this register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCEN**: Delegated configuration enable

0: RIF configuration for subregion A can be written only by the trusted domain.

1: RIF configuration for subregion A can be written only by compartment DCCID, defined in this register. If base region x is defined as secure (SEC = 1), nonsecure write accesses to this subregion RIF configuration by compartment DCCID are ignored.

Note: RIF configuration of subregion A is defined as RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR.

Bits 1:0 Reserved, must be kept at reset value.

10.5.16 RISAF region x subregion B configuration register (RISAF_REGx_BCFGR)

Address offset: 0x060 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 in this register.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register.

When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC[1:0]		WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID[2:0]			Res.	Res.	RLOCK	SREN
r	r	rw	rw			rw	rw		rw	rw	rw			rs	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ENC[1:0]**: Encrypted subregion

This read-only bitfield is a mirror to ENC[1:0] in RISAF_REGx_CFGR. It is reserved if the encryption feature is not available in RISAF (see ENCDIS in RISAF_SR, and *Section 10.3*).

0x0: No effect

0x1, 0x2: Allowed read (write) requests to subregion B of region x are decrypted (encrypted)

0x3: No effect

Bit 13 **WREN**: Write enable

This bit is taken into account only if SREN = 1 in this register.

0: Any write to subregion B of region x is ignored

1: Subregion B of region x can be written

Bit 12 **RDEN**: Read enable

This bit is taken into account only if SREN = 1 in this register.

0: Any read to subregion B of region x returns 0

1: Subregion B of region x can be read

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged subregion

This bit is taken into account only if SREN = 1 in this register.

0: Privileged and unprivileged accesses are granted in subregion B of region x

1: Only privileged accesses are granted in subregion B of region x

Note: When base region x is defined as unprivileged for the SCID compartment (PRIVCy = 0 in RISAF_REGx_CFGR, with SRCID = Cy), unprivileged accesses to subregion B are always granted.

Bit 8 **SEC**: Secure subregion

This bit is taken into account only if SREN = 1 in this register. Nonsecure modification to it are discarded.

0: Only nonsecure requests can access subregion B of region x

1: Only secure requests can access subregion B of region x

Note: When base region x is defined as nonsecure (SEC = 0 in RISAF_REGx_CFGR), only nonsecure accesses to subregion B are granted.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SRCID[2:0]**: Subregion CID

When SREN = 1 in this register, SRCID[2:0] defines which compartment has access to subregion B of RISAF region x.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RLOCK**: Resource lock

When this bit is set, the subregion resource is locked.

0: RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR are writable

1: Writes to RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR are ignored

Note: This bit is set once, cleared only by RISAF reset. It cannot be set if GLOCK = 0 in RISAF_CR.

Bit 0 **SREN**: Subregion enable

0: Subregion B of region x is disabled. Access control of base region x applies to any access between this subregion start and end addresses.

1: Subregion B of region x is enabled. Access control defined in subregion B applies to any access between subregion B start and end addresses.

Note: The filtering starts from address RISAF_REGx_BSTARTR (included), and ends at address RISAF_REGx_BENDR (included).

10.5.17 RISAF region x subregion B start-address register (RISAF_REGx_BSTARTR)

Address offset: 0x064 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_BCFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register. When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADDSTART[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDSTART[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SADDSTART[31:0]**: Subregion address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in subregion B in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDSTART below the associated BADDSTART).

10.5.18 RISAF region x subregion B end-address register (RISAF_REGx_BENDR)

Address offset: 0x068 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

This reset value applies when the granularity is 4 Kbytes. More in general, if the granularity is 2^n, the reset value is 2^n - 1.

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_BCFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register. When the delegation is disabled (DCEN = 0), only the trusted domain can program this register.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADDEND[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDEND[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SADDEND[31:0]**: Subregion address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address of subregion B in base region x.

Address bits less than the granularity cannot be modified, and remain 1.

Address bits greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDEND higher than the associated BADDEND).

10.5.19 RISAF region x subregion B nested mode register (RISAF_REGx_BNESTR)

Address offset: 0x06C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 10.3* to verify the number of base regions and subregions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID[2:0]			Res.	DCEN	Res.	Res.
									rw	rw	rw		rw		

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **DCCID[2:0]**: Delegated configuration CID

The trusted domain can use this bitfield to statically define which compartment can program the RIF configuration of subregion B if DCEN = 1 in this register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCEN**: Delegated configuration enable

0: RIF configuration for subregion B can be written only by the trusted domain.

1: RIF configuration for subregion B can be written only by compartment DCCID, defined in this register. If base region x is defined as secure (SEC = 1), nonsecure write accesses to this subregion RIF configuration by compartment DCCID are ignored.

Note: RIF configuration of subregion B is defined as RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR.

Bits 1:0 Reserved, must be kept at reset value.

10.5.20 RISAF hardware configuration register (RISAF_HWCFGR)

Address offset: 0xFF0

Reset value: 0xXXXX XXXX

Refer to *Section 10.3* for each instance reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG4[7:0]								CFG3[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								CFG1[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **CFG4[7:0]**: Hardware generic 4

This bitfield returns the number of address bits to match when determining access to a base region or subregion (WIDTH).

Bits 23:16 **CFG3[7:0]**: Hardware generic 3

This bitfield returns the smallest possible region size (granularity) (LOG_BLOCKSIZE).

Bits 15:8 **CFG2[7:0]**: Hardware generic 2

This bitfield returns the total number of subregions A and B (NUM_SUBREGIONAS, NUM_SUBREGIONBS).

Bits 7:0 **CFG1[7:0]**: Hardware generic 1

This bitfield returns the number of base regions (NUM_REGS).

10.5.21 RISAF version register (RISAF_VERR)

Address offset: 0xFF4

Reset value: 0xXXXX XXXX

Refer to *Section 10.3* for each instance reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**:RISAF major revision

Bits 3:0 **MINREV[3:0]**: RISAF minor revision

10.5.22 RISAF identification register (RISAF_IPIDR)

Address offset: 0xFF8

Reset value: 0xXXXX XXXX (refer to *Section 10.3* for each instance reset value)

This offset is valid for instances not supporting the DDRMCE feature. For instance with DDRMCE feature, the offset is 0x1FF8.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RISAF identification code

10.5.23 RISAF size identification register (RISAF_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD0X

This offset is valid for instances not supporting the DDRMCE feature. For instance with DDRMCE feature, the offset is 0x1FFC. Refer to *Section 10.3: RISAF implementation* for each instance reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RISAF size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 or 0x08 (1- or 8-Kbyte address decoding)

10.5.24 RISAF extended configuration register (RISAF_XCR)

Address offset: 0x1C00

Reset value: 0x0000 0100

Secure, privileged, trusted domain TDCID, write access only. Writes are ignored if XLOCK is set in RISAF_XCR.

This register is reserved if the DDRMCE feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIPHERSEL[1:0]		Res.	Res.	MKLOCK	XLOCK
										rw	rw			rs	rs

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **CIPHERSEL[1:0]**: Cipher selection

Set these bits to select the encryption algorithm for encrypted regions with ENC[1:0] = 0x1. When CIPHERSEL[1:0] = 0x0, any read (resp. write) to an encrypted region with BREN = 1 and ENC[1:0] = 0x1 returns 0 (resp. is ignored).

0x0: No cipher selected

0x1, 0x2: AES-128 cipher selected

0x3: AES-256 cipher selected

When CIPHERSEL value is changed with a successful write, RISAF_MKEYRx registers are 0-ed.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **MKLOCK**: Master keys lock

Lock the master key configuration until next reset. This bit is cleared by default, and, once set, cannot be reset until MCE reset.

0: Writes to RISAF_MKEYRy registers are is allowed

1: Writes to RISAF_MKEYRy registers are is ignored until next RISAF reset

Bit 0 **XLOCK**: Extended lock

Set to lock the extended configuration of RISAF (RISAF_XCR and RISAF_MKEYRy registers). This bit is cleared by default, and, once set, cannot be reset until RISAF reset.

0: Extended registers are writable

1: All writes to extended registers are ignored

10.5.25 RISAF extended status register (RISAF_XSR)

Address offset: 0x1C04

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only.

This register is reserved if the DDRMCE feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MKVALID
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MKVALID**: Master key valid

0: A valid key has not been written in RISAF_MKEYRy registers, or a valid unlocked master key has been erased by hardware (if bit was set before).

1: A valid key has been written in RISAF_MKEYRy registers (write R0 → R1 → R2 → R3, or the reverse).

When this bit is set, any successful write to RISAF_MKEYRy registers clears MKVALID bit (for example after a write when XLOCK or MKLOCK are cleared).

10.5.26 RISAF master key y (RISAF_MKEYRy)

Address offset: 0x1E00 + 0x4 * y (y = 0 to 7)

Reset value: 0x0000 0000

Secure, privileged, trusted domain TDCID, read access only. Writes are ignored if XLOCK is set in RISAF_XCR.

These register are reserved if the DDRMCE feature is not available in the RISAF instance (see *Section 10.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MKEY{i + 32 * y}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MKEY{i + 32 * y}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **MKEY{i + 32 * y}**: Cryptographic key (i = 31 to 0)
This write-only bitfield contains the key used by the DDRMCE when ENC[1:0] = 0x1 (side-channel protected mode).

10.5.27 RISAF register map

Table 50. RISAF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RISAF_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x004	RISAF_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENCDIS	KEYRDY	KEYVALID
	Reset value																														0	0	0
0x008	RISAF_IASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF1	IAEF0	CAEF
	Reset value																														0	0	0
0x00C	RISAF_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AEF1	AEF0	CAEF
	Reset value																														0	0	0
0x010-0x01C	Reserved	Reserved																															
0x020 + 0x8 * x (x=0 to 1)	RISAF_IAESRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IANRW	Res.	IASEC	IAPRIV	Res.	IACID [2:0]		
	Reset value																									0		0	0		0	0	0
0x024 + 0x8 * x (x=0 to 1)	RISAF_IADDR	IADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030 + 0x4 * x (x=0 to 3) Last address: 0x03C	RISAF_KEYRx	KEY{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50. RISAF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040 + 0x40 * (x-1) (x=1 to 15) Last address: 0x3C0	RISAF_REGx_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVC7	PRIVC6	PRIVC5	PRIVC4	PRIVC3	PRIVC2	PRIVC1	PRIVC0	ENC[1]	ENC[0]	Res.	Res.	Res.	Res.	Res.	SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
	Reset value									0	0	0	0	0	0	0	0	0	0						0								0
0x044 + 0x40*(x-1) (x=1 to 15) Last address: 0x3C4	RISAF_REGx_STARTR	BADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048 + 0x40*(x-1) (x=1 to 15) Last address: 0x3C8	RISAF_REGx_ENDR	BADDEND[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04C + 0x40*(x-1) (x=1 to 15) Last address: 0x3CC	RISAF_REGx_CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRENC7	WRENC6	WRENC5	WRENC4	WRENC3	WRENC2	WRENC1	WRENC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDENC7	RDENC6	RDENC5	RDENC4	RDENC3	RDENC2	RDENC1	RDENC0
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x050 + 0x40*x - 1) (x=1 to 15) Last address: 0x3D0	RISAF_REGx_ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC[1]	ENC[0]	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID [2:0]			Res.	Res.	RLOCK	SREN
	Reset value																	0	0	0	0			0	0		0	0	0			0	0
0x054 + 0x40*(x-1) (x=1 to 15) Last address: 0x3D4	RISAF_REGx_ASTARTR	SADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058 + 0x40*(x-1) (x=1 to 15) Last address: 0x3D8	RISAF_REGx_AENDR	SADDEND[31:0]																															
	Reset value(3)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x05C + 0x40*(x-1) (x=1 to 15) Last address: 0x3DC	RISAF_REGx_ANESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID [2:0]			Res.	DCEN	Res.	Res.
	Reset value																										0	0	0		0		
0x060 + 0x40*(x-1) (x=1 to 15) Last address: 0x3E0	RISAF_REGx_BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC[1]	ENC[0]	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID [2:0]			Res.	Res.	RLOCK	SREN
	Reset value																	0	0	0	0			0	0		0	0	0			0	0
0x064 + 0x40*(x-1) (x=1 to 15) Last address: 0x3E4	RISAF_REGx_BSTARTR	SADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068 + 0x040*(x-1) (x=1 to 15) Last address: 0x3E8	RISAF_REGx_BENDR	SADDEND[31:0]																															
	Reset value(3)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 50. RISAF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x06C + 0x40*(x-1) (x=1 to 15) Last address: 0x3EC	RISAF_REGx_BNESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID [2:0]			Res.	DCEN	Res.	Res.
	Reset value																										0	0	0		0		
0x1C00(2)	RISAF_XCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIPHER [1:0]		Res.	Res.	MKLOCK	XLOCK
	Reset value																											0	0			0	0
0x1C04(2)	RISAF_XSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MKVALID
	Reset value																																0
0x1E00 + 0x4 * x, (x=0 to 7) Last address: 0x1E1C(2)	RISAF_MKEYRx	MKEY{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3F0-0xFEC	Reserved	Reserved																															
0xFF0	RISAF_HWCFGR	CFG4[7:0]								CFG3[7:0]								CFG2[7:0]								CFG1[7:0]							
	Reset value(3)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFF4	RISAF_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]		
	Reset value(3)																										X	X	X	X	X	X	X
0xFF8(4)	RISAF_IPIDR	ID[31:0]																															
	Reset value(3)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	X
0xFFC(4)	RISAF_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0
0x1FF8(2)	RISAF_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	1
0x1FFC(2)	RISAF_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	1	0	0	0

1. Reset value is 0x0000 0FFF when the granularity is 4 Kbytes.
2. This register is not reserved if this RISAF instance supports the DDRMCE option.
3. Refer to *Section 10.3: RISAF implementation* for the actual reset value.
4. This register is reserved if this RISAF instance supports the DDRMCE option.

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

11 Illegal access controller (IAC)

11.1 IAC introduction

The RIF (resource isolation framework) is a comprehensive set of hardware blocks designed to enforce and manage the isolation of STM32 hardware resources, such as memory and peripherals.

The RIF uses the IAC (illegal access controller) to centralize the detection of RIF-related illegal accesses, managed by a secure application running in the trusted domain.

11.2 IAC main features

- Centralized detection of RIF-related illegal accesses, managed by a secure application running in the trusted domain. Supported illegal accesses are: secure, privileged, CID.
- Only accessible by secure privileged trusted domain CPU

11.3 IAC implementation

The index table (from 0 to 127) of the non-RIF-aware peripherals managed by the IAC is centralized in *Section 8: Resource isolation framework security controller (RIFSC)*.

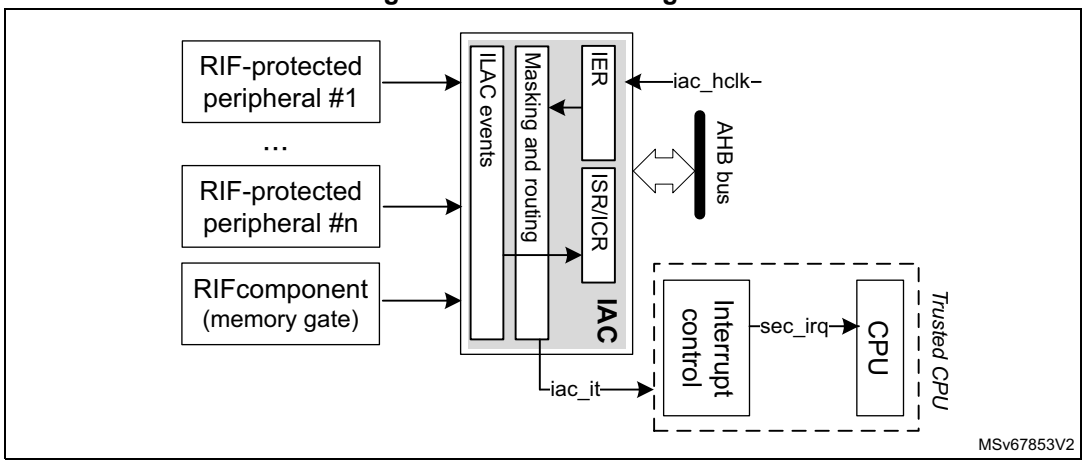
The index table (from 128 and above) for the RIF components and RIF-aware peripherals also managed by the IAC is found as well in *Section 8: Resource isolation framework security controller (RIFSC)*.

11.4 IAC functional description

11.4.1 IAC block diagram

Figure 32 shows the IAC block diagram, in context.

Figure 32. IAC block diagram



11.4.2 IAC internal signals

Table 51 lists the internal signals available at IAC level, not at product level.

Table 51. IAC internal signals

Name	Type	Description
iac_hclk	Digital input	AHB bus clock
iac_it	Digital output	Global interrupt request

11.4.3 IAC reset and clocks

The IAC peripheral clock is automatically managed by the device, and the peripheral is reset only by a system reset.

The IAC requires its clock in order to trigger an interrupt request event.

11.4.4 IAC use in RIF

The IAC gathers illegal events generated within the system when an illegal access is detected. The IAC can then generate a secure interrupt towards the secure trusted CPU, if needed. By default, all events in the IAC are masked.

The IAC has room for 256 interrupt sources. Each source corresponds to an IAC index from 0 to 255. The interrupt sources from non-RIF-aware peripherals occupy most of the first 128 indexes. The interrupt sources from RIF-aware peripherals and RIF firewalls occupy indexes from 128 onwards. All these sources are defined in *Section 11.3*.

Refer to *Section 5.4: Resource isolation framework (RIF)* for more information on illegal access definitions.

11.4.5 IAC management by trusted application

For each interrupt source of index <i>, an illegal access sets the IAFi flag in IAC_ISRx. A secure privileged trusted application can clear this flag using the corresponding bit in IAC_ICRx. An IAC interrupt is triggered when IAFi is set while IAIEi is set in IAC_IERx.

Indexes between 0 to 127 are assigned to non-RIF-aware peripherals, with a RISUP block instantiated in front of its configuration port to filter configuration accesses. The programming of these RISUP blocks (in the RIFSC) uses the same index as the IAC.

Indexes 128 and above are assigned to RIF components and RIF-aware peripherals. These peripherals can block illegal accesses to their configuration registers (without RISUP).

For RIF-aware peripherals that protect a memory, a secure privileged trusted application can obtain, in the peripheral, more details about the cause of the illegal access event:

- The illegal access status register (xx_IASR) indicates if it is a configuration access error (CAEF= 1), or/and a system bus access error (IAEF = 1).
- The illegal access error status register (xx_IAESR) and the illegal address register (xx_IADDR) give detailed information on the system bus access error.

The IAC interrupt handler must clear IAFi in IAC_ISR before clearing the error flag(s) in the firewall peripheral xx (using xx_IACR register). Otherwise, an illegal access can occur between clearing the flags in the firewall and clearing the flag in the IAC, and the application is unaware of this error.

11.5 IAC interrupts

The RIF interrupt management in the IAC is summarized in *Table 52*.

Note: Only secure privileged trusted domain application, or debugger executing in the secure privileged debug domain can enable, read, and clear illegal access interrupts.

If there is an illegal access to the IAC itself, the status flag corresponding to the IAC index in Section 8: Resource isolation framework security controller (RIFSC) is set.

Table 52. IAC interrupt request

Acronym	Interrupt event	Event flag in IAC_ISR	Enable control bit in IAC_IER	Interrupt clear method
IAC	Illegal access error	IAF	IAIE	Set IAF bit in IAC_ICR

11.6 IAC registers

All IAC registers are accessed only by words (32 bits).

If the debug domain feature is activated and security and privileged policy allows it, the debug domain traffic goes through, regardless of any CID filtering policy. If the debug domain feature is de-activated the CID filtering policy also applies to debug domain traffic.

11.6.1 IAC interrupt enable register x (IAC_IERx)

Address offset: 0x000 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID, read and write only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAIE{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAIE{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IAIE{i + 32 * x}**: Illegal access interrupt enable for peripheral {i + 32 * x} (i = 0 to 31)
Each bit is set to unmask illegal access events from peripheral {i + 32 * x}.
0: Illegal access event from peripheral {i + 32 * x} does not generate interrupt (masked).
1: Illegal access event from peripheral {i + 32 * x} can generate interrupts (unmasked).

11.6.2 IAC interrupt status register x (IAC_ISRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID, read only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IAF{i + 32 * x}**: Illegal access interrupt enable for peripheral {i + 32 * x} (i = 0 to 31)

Each bit is set when an illegal access event occurs in the peripheral {i + 32 * x} (see *Section 11.3* for details). This bit is cleared when the corresponding IAF bit is set in IAC_ICRx.

0: No illegal access event detected for peripheral {i + 32 * x} (since reset or the last time this bit was cleared).

1: At least one illegal access event has been detected for peripheral {i + 32 * x} (since the last time this bit was cleared).

11.6.3 IAC interrupt clear register x (IAC_ICRx)

Address offset: 0x100 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, trusted domain CID, write only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAF{i + 32 * x}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAF{i + 32 * x}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **IAF{i + 32 * x}**: Illegal access flag clear for peripheral {i + 32 * x} (i = 0 to 31)

Setting each bit clears the status flag of the illegal access event {i + 32 * x} in IAC_ISRx.

0: IAF {i + 32 * x} flag status not affected

1: IAF {i + 32 * x} flag status cleared in IAC_ISRx

11.6.4 IAC ILAC input status register x (IAC_IISRx)

Address offset: 0x36C + 0x4 * x (x = 0 to 5)

Reset value: 0xCFFE FFFF, 0xBFDE 0F3F, 0xF9C3 760C, 0xA1DC 7DFF, 0xFFB9 12D7, 0x0003 C9FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ILACIN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILACIN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ILACIN{i + 32 * x}**: Illegal access input {i + 32 * x} (i = 0 to 31)
0: ILAC input {i + 32 * x} to IAC not present
1: ILAC input {i + 32 * x} to IAC present

11.6.5 IAC hardware configuration register 2 (IAC_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CFG2[3:0]**: Hardware configuration 2
This bitfield returns the generic value DEFAULT_IER.

Bits 3:0 **CFG1[3:0]**: Hardware configuration 1
This bitfield returns the generic value NUM_RESYNC_FLOPS.

11.6.6 IAC hardware configuration register 1 (IAC_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x00B4 3111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG5[8:0]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **CFG5[8:0]**: Hardware configuration 5
This bitfield returns the generic value NUM_ILAC.

Bits 15:12 **CFG4[3:0]**: Hardware configuration 4
This bitfield returns the generic value CID_WIDTH.

Bits 11:8 **CFG3[3:0]**: Hardware configuration 3
This bitfield returns the generic value PRIV_ENABLED.

Bits 7:4 **CFG2[3:0]**: Hardware configuration 2
This bitfield returns the generic value SEC_ENABLED.

Bits 3:0 **CFG1[3:0]**: Hardware configuration 1
This bitfield returns the generic value RIF_ENABLED.

11.6.7 IAC version register (IAC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**:IAC major revision

Bits 3:0 **MINREV[3:0]**: IAC minor revision

11.6.8 IAC identification register (IAC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0163

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IAC identification code

11.6.9 IAC size identification register (IAC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: IAC size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1-Kbyte address decoding)

11.6.10 IAC register map

Table 53. IAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 +0x4*x (x = 0 to 5) Last address: 0x014	IAC_IERx	IAIE{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x07C	Reserved	Reserved																															
0x080 +0x4*x (x = 0 to 5) Last address: 0x094	IAC_ISRx	IAF{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098-0x0FC	Reserved	Reserved																															
0x100 +0x4*x (x = 0 to 5) Last address: 0x114	IAC_ICRx	IAF{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118-0x368	Reserved	Reserved																															
0x36C +0x4*x (x = 0 to 5) Last address: 0x384	IAC_IISRx	ILACIN{i + 32 * x}																															
	Reset value	0xCFFE FFFF, 0xBFDE 0F3F, 0xF9C3 760C, 0xA1DC 7DFF, 0xFFB9 12D7, 0x0003 C9FF																															
0x388-0x3E8	Reserved	Reserved																															
0x3EC	IAC_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
	Reset value																									0	0	0	0	0	0	1	0
0x3F0	IAC_HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG5[8:0]									CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value								0	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1
0x3F4	IAC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	IAC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1
0x3FC	IAC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

12 DDR subsystem architecture (DDRSS)

12.1 DDRS introduction

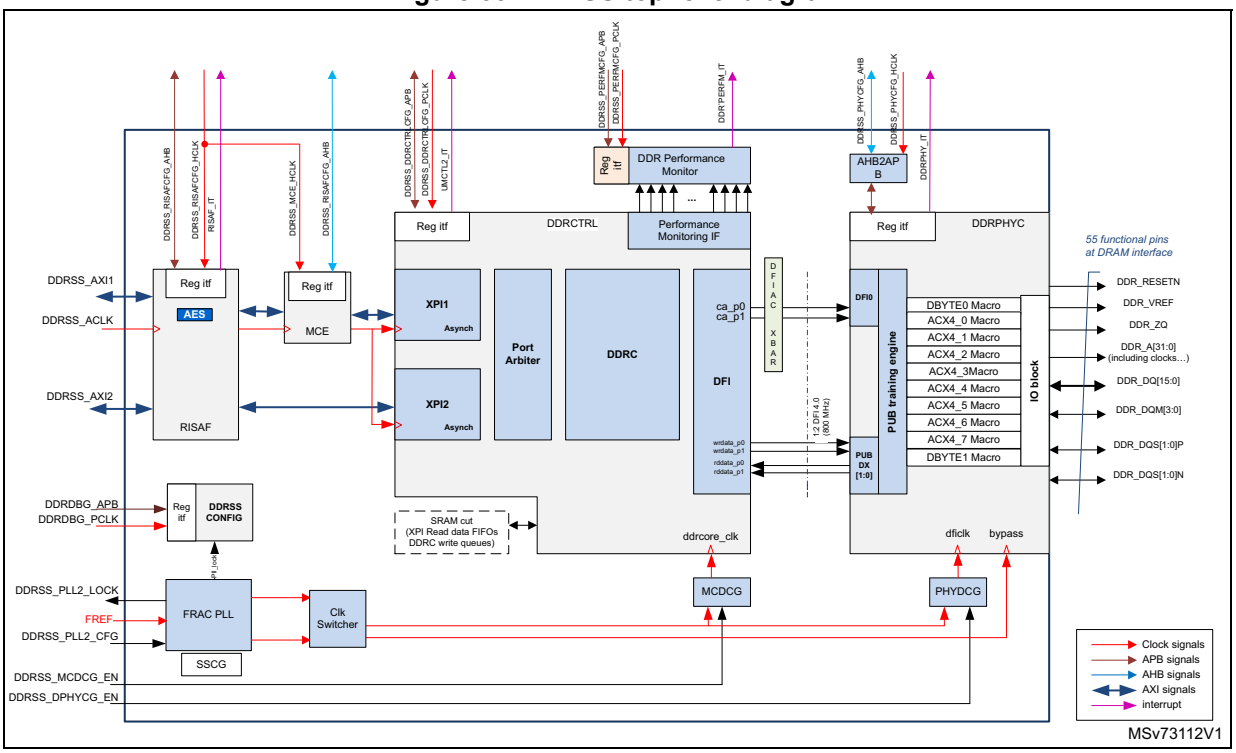
The DDR subsystem (DDRSS) includes:

- Part of the RIF infrastructure, a dual-port RISAF configurable filtering engine with 128-bit AES encryption/decryption capability.
- A Memory cipher engine (MCE) with AES 128/256-bit encryption/decryption features including leakage resilient mode of operation as defense against side channel.
- A dual-port DDRCTRL (see *Section 13: DDR3L/DDR4/LPDDR4 controller (DDRCTRL)*) that gathers all system requests, arbitrates, and schedules the DDR bus accesses.
- A DDRPHYC (see *Section 14: DDR physical interface control (DDRPHYC)*) responsible for adapting DDR command to one of the JEDEC DDR standards (LPDDR4, DDR3L or DDR4).
- A local widely configurable PLL2 generating the clock for the DDRCTRL core part and the DDRPHYC, plus the clock forwarded by the DDRPHYC to the external SDRAM.
- Embedded top-level registers (DDRDBG) for configuring some DDRSS top-level glue logic features, such as DDR3/4 AC swizzling, low-power features, mode register read data accesses, and observability signal muxing.
- A programmable performance monitor DDRPERFM (see *Section 15: DDR performance monitor (DDRPERFM)*) that monitors the DFI interface and reports selected performances.

Caution: DDRCTRL and DDRPHYC are configured to support multiple ranks (dual rank), however it is strongly recommended to use single rank LPDDR4 devices.
Dual rank implementation is susceptible to signal integrity (SI) limitations due to unterminated lanes inside the DRAM which can restrict its frequency range.

12.2 DDRSS block diagram

Figure 33. DDRSS top-level diagram



12.3 DDRSS interfaces

The DDRSS connects to the system bus through the following interfaces:

- Two AXI4 ports @ 600 MHz for actual DDR SDRAM data accesses.

  Only one of these ports (port 0) goes through the AES encryption/decryption in the RISAF (not present on all devices, refer to the datasheet).
- One APB interface @200 MHz for accessing the DDRCTRL registers.
- One APB interface @200 MHz for accessing the DDRPERFM registers.
- One APB interface @200 MHz for accessing/programming top-level DDRSS registers.

  Through this interface, the following features are programmed: low-power (partially), observability, AC-swizzling for DDR3/DDR4, mode register read value access, and data-rate adaptation.
- One AHB interface @150 MHz for accessing/programming the RISAF4 registers, its filtering regions, and which ones are encrypted.
- One AHB interface @150 MHz for accessing/programming the MCE registers.
- One AHB interface @150 MHz for accessing/programming DDRPHYC registers.

Note: The RIF protection is detailed in Section 20: Reset and clock control (RCC) (not RIFSC) and pointer towards RCC (for access rights and low-power status), and Section 5.4: Resource isolation framework (RIF) (pointer to RIF) for access rights.

12.4 DDRSS embedded fractional PLL2

The DDRSS embeds locally a fractional PLL2 provided with a spread-spectrum modulator. While physically located in the DDRSS, this PLL2 is under the RCC control. Especially, registers controlling the different PLL2 tuning are RCC registers.

The clock delivered by the PLL2 is configurable in a range from 16 MHz to 3200 MHz, and addresses the DDRCTRL core part and the DDRPHYC. The DDRPHYC relies on this clock for the DDR interface transactions.

The spread-spectrum modulator connected to the PLL2 allows modulating the DRAM clock frequency (triangular down-spread modulation) in order to reduce EMI at DRAM interface.

Depending on the requested data rate, the clock distribution to the DDRPHYC varies as follows.

- for data rates greater than 667 MT/s:
  - Clear the DDRDBG_BYPASS_PCLKEN register to 0 (BypassPclk is gated low).
  - Set the PLL2 to deliver a frequency of data rate/4 MHz.
  - DDRCTRL and DDRPHYC operate synchronously at half of the DRAM clock frequency.
  - A low-jitter PLL embedded in the DDRPHYC delivers a high-speed DDRPHYC internal clock running at four times the DDRPHYC input clock frequency. The DDRPHYC delivers also the DRAM differential clocks (DDRA_CKP/N and DDRB_CKP/N) running at two times the DDRPHYC input clock frequency.
- for data rates lower than 667 MT/s:
  - Set the DDRDBG_BYPASS_PCLKEN register to 1 (BypassPclk is active).
  - Set the PLL2 to deliver a frequency of data rate MHz.
  - The DDRCTRL and DDRPHYC DFI interfaces operate synchronously at data rate/4 MHz.
  - BypassPclk clock runs at four times the DFI interface clock. BypassPclk connects to the PLL2 main output, and DFI_clock/ddrc_core_clock connect to the “divided-by-4” output.

Note: Enabling BypassPclk_enable also drives to 1 the PLL2 Foutpostdiven input to enable the FOUT2 output.

The table below presents, depending on the configured data rate, the various frequency values in the DDRSS.

Table 54. Frequency values for the DDRSS clocks versus data rate versus DDR JEDEC standard

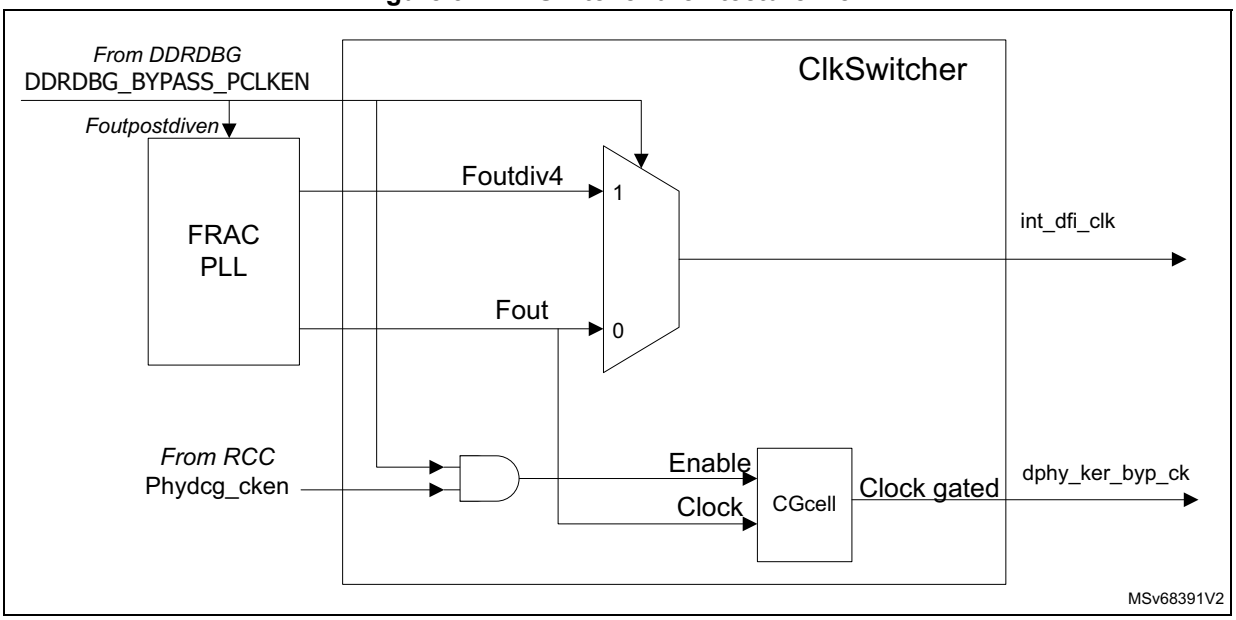
Conditions		Bypass Pclk_ enable	PLL2 delivered clock (MHz)	PLL2 divided by 4 delivered clock (MHz)	DDRCTRL core clock (MHz)	DDRPHYC DFI clock (MHz)	DDRPHYC bypass Pclk (MHz)	DDR_CLK (MHz)
Data rate < 667 MT/s	LPDDR4	1	Up to 666	Up to 166	Up to 166	Up to 166	Up to 166	Up to 333
	DDR3L	1	Up to 666	Up to 166	Up to 166	Up to 166	Up to 166	Up to 333
	DDR4	DDR4 DLL-on mode only possible starting from 625 MHz						

Table 54. Frequency values for the DDRSS clocks versus data rate versus DDR JEDEC standard (continued)

Conditions		Bypass Pclk_ enable	PLL2 delivered clock (MHz)	PLL2 divided by 4 delivered clock (MHz)	DDRCTRL core clock (MHz)	DDRPHYC DFI clock (MHz)	DDRPHYC bypass Pclk (MHz)	DDR_CLK (MHz)
Data rate > 667 MT/s	LPDDR4	0	Up to 400	Gated down	Up to 400	Up to 400	Gated down	Up to 800
	DDR3L	0	Up to 400	Gated down	Up to 400	Up to 400	Gated down	Up to 800
	DDR4	0	Up to 400	Gated down	Up to 400	Up to 400	Gated down	Up to 800

The figure below presents a high-level view of the ClkSwitcher logic.

Figure 34. ClkSwitcher architecture view



12.5 DDRSS top-level features

12.5.1 Filtering and AES encryption

Incoming data transfers to the DDRSS are first checked against security rules. The RISAF4 performs this security checking. The security rules are defined per SDRAM memory subregions, and programmed into the RISAF through its AHB interface (see *Section 10.5: RISAF registers* more details on programming rules).

The DDRSS is dual port, as the RISAF and the DDRCTRL. The security rules are set identically for all these access ports.

The read transactions that do not comply to the security rules result as read-as-zeros (RAZ).

The write transactions that do not comply to the defined security rules result in non-effective writes: the write transaction is forwarded downstream but nothing is written in the SDRAM array.

At reset, the RISAF is configured with a single region covering the whole SDRAM memory space, and accepting only trusted domain (secure and privilege) traffic.

On top of security defined rules, some regions can be AES-128 encrypted. Encrypted regions are programmed through the RISAF registers. The AES encryption/decryption engine is embedded in the RISAF. While the RISAF is dual port, the AES encryption/decryption engine can be applied only on RISAF port 1.

In addition, memory region can be defined to be encrypted using MCE with AES-128 bits or AES 256 bits encryption/decryption engines. MCE provides advanced features against side channel attacks.

The STM32MP21xx bus interconnect is set so that the following initiator list is candidate for encryption: CPU1, CPU2, HPDMA1/2/3 AHB traffic, HPDMA 1/2/3 AXI traffic, and LTDC secure traffic.

At DDRSS initialization time, the 128-bit encryption key must be written in the RISAF registers before any data transfer occurs.

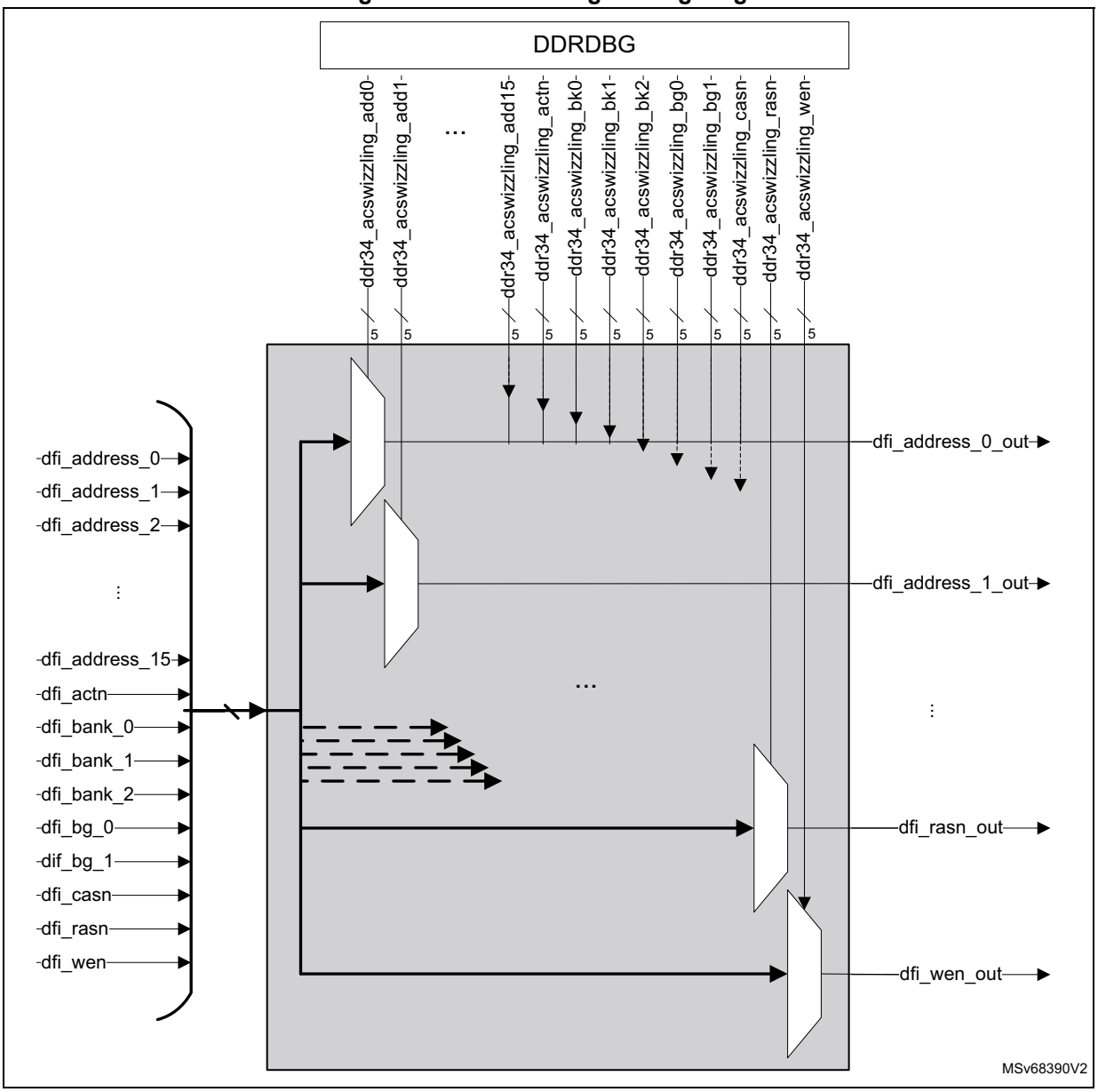
12.5.2 DDR3/4 AC-swizzling

The DDRPHYC natively allows swapping address pad connections for LPDDR4. This is especially helpful when the PCB routing requires to swap the AC ball connections (see *Section 14.2: DDRPHYC block diagram* for more details).

The DDRPHYC does not natively support the same feature for DDR3 and DDR4.
At the DFI interface between the DDRCTRL and the DDRPHYC, a set of multiplexers enables this functionality. These multiplexers driving inputs are programmable through DDRSS DDRDBG registers. The figure below presents this muxing stage.

Any of the 25 input signals can be swapped to match the remapping performed downstream of the DDRPHYC. By default, these signals are not remapped.

Figure 35. AC-swizzling muxing stage



12.5.3 Mode register read

The LPDDR4 standard is provided with a mode register read command. The DDR4 is provided with a multi-purpose register read command. The DDRCTRL supports these commands but does not register the resulting read value.

The DDRSS is then enhanced with registers to store these values, to monitor the status, and to clear them.

Addresses to these registers are the following:

- MRR 0x4804 2000 - 128-bit data (RO)
- MRR 0x4804 2010 - status (RO)
- MRR 0x4804 2014 - clear (WO)

The sequence to perform a mode read register (LPDDR4) or a multi-purpose register read (DDR4) is detailed below:

1. Program the mode register read through the DDRCTRL APB interface:
   a) The DDRCTL performs the command sequence for reading the mode register value.
   b) When the read value comes back on the hif_mrr_data interface and is valid, it is stored in the DDRDBG MRR read value.
   c) The DDRDBG MRR status indicates a valid data is present in the register.
2. Poll on the DDRDBG MRR status register through the DDRSS DDRDBG APB interface.
3. When the DDRDBG MRR status register indicates a valid data is present in the read value register, read it from the DDRDBG APB interface.
4. Clear the read data and the status bit writing the clear register bit through the DDRDBG APB interface.

12.5.4 Observability

The DDRSS reports 16 parallel signals to the HDP (hardware debug port):

- Twelve of these signals are directly tied to DDRSS internal signals.
- Four of these signals are software controllable (through DDRSS DDRDBG) to select four out of a max of 16 meaningful signals.

The 12 signals directly assigned to an observability port of the HDP are listed in the following table.

Table 55. List of observable signals sent directly to HDP

#	Signal	Source	Meaning
0	ddrss_lpiddrc_csysreq	RCC	csysreq signal from the DDRCTRL core low-power interface
1	ddrss_lpiddrc_csysack	DDRCTRL	csysack signal from the DDRCTRL core low-power interface
2	ddrss_lpiddrc_cactive	DDRCTRL	cactive signal from the DDRCTRL core low power interface
3	ddrss_mcdcg_en	RCC	cken signal from the RCC to the MCDCG block
4	ddrss_dphycg_en	RCC	cken signal from the RCC to the PHYDCG block
5	dfi0_init_start	DDRCTRL	dfi interface signal, used for initialization and going into LP2/LP3 low-power mode
6	dfi0_init_complete	DDRPHYC	dfi interface signal, used for initialization and going into LP2/LP3 low-power mode
7	dfi0_ctrlupd_req	DDRCTRL	Update request from the controller that the DDRPHYC updates the delay-line values from the master delay line
8	dfi0_phyupd_req	DDRPHYC	DDRPHYC requests to the DDRCTRL to hold traffic for the time the DDRPHYC performs a VT compensation of its internal delay line.
9	dfi0_phymstr_req	DDRPHYC	DDRPHYC requests control over the DRAM bus.
10	dfi0_lp_req	DDRCTRL	There is no opportunity for low-power mode entry.
11	dfi0_freq[0]	DDRCTRL	Used, in STM32MP21xx, to switch to LP2/LP3 low-power mode

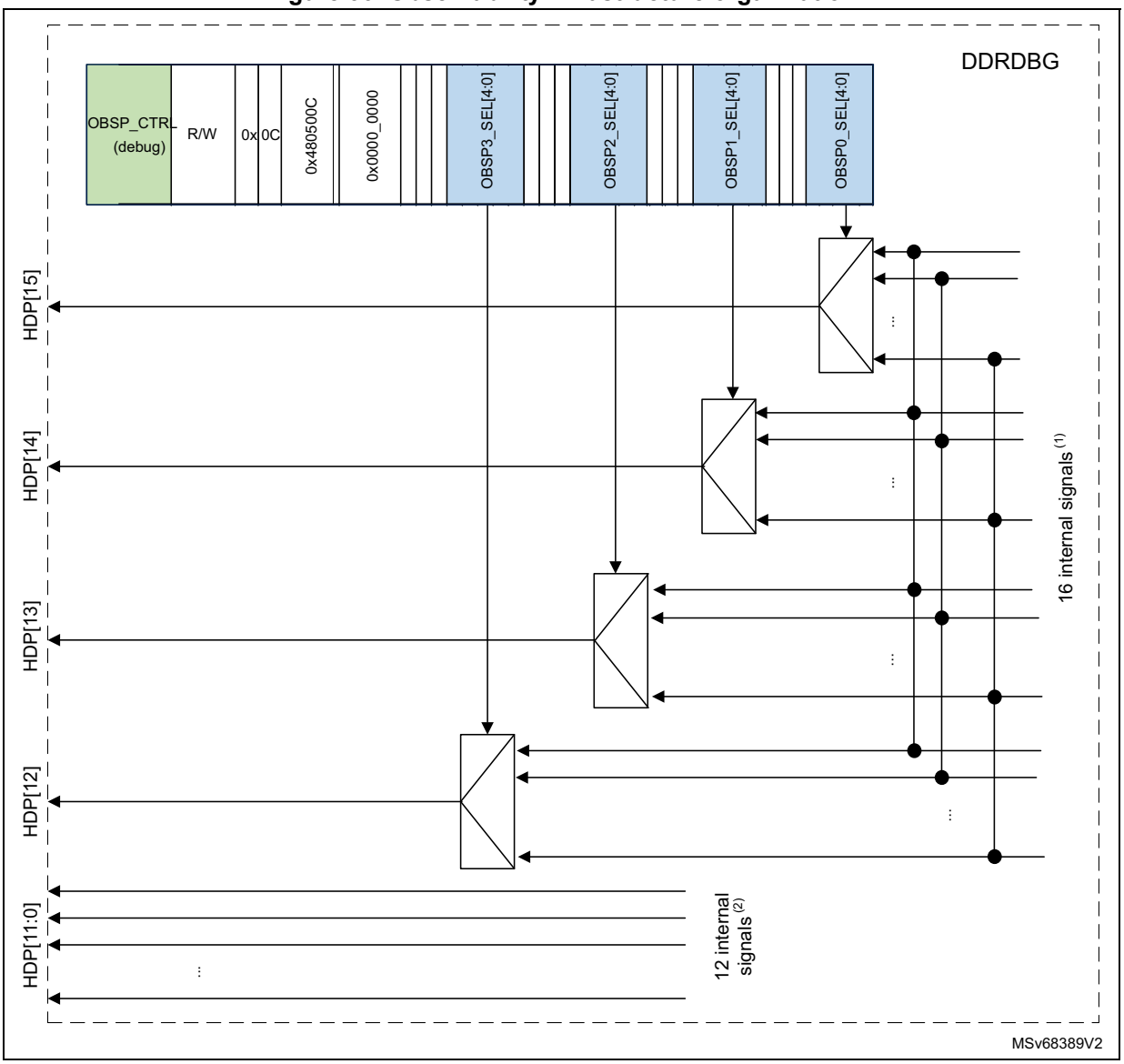
The pool of 16 signals among which four can be selected to be routed to the HDP are listed in the table below.

Table 56. List of observable signals selectable and sent to HDP

#	Signal	Source	Meaning
0	ddrctrl_reg_selfref_type[0]	DDRCTRL	Bit 0 of the reg_selfref_type DDRCTRL register
1	ddrctrl_reg_selfref_type[1]	DDRCTRL	Bit 1 of the reg_selfref_type DDRCTRL register
2	pll2_div_10	Frac_PLL	Main output of the Frac_PLL, divided by 10
3	pll2_lock	Frac_PLL	Lock of the frac_PLL
4	dwc_ddrphy_dto_div_10	DDRPHYC	DDRPHYC internal PLL output divided by 10
5	dwc_ddrphy_dto	DDRPHYC	DDRPHYC internal PLL lock
6	lpi_cactive_n_1	DDRCTRL	cactive from AXI port 1, allowing to shut down port clock
7	lpi_cactive_n_2	DDRCTRL	cactive from AXI port 2, allowing to shut down port clock
8	ddrss_phy_pwok_tcr	PWRCTRL	PWROKIN provided to the DDRPHYC by the PWRCTRL
9	um_ddrss_axi1_awpoison and um_ddrss_axi1_awvalid	RISAF	At the output of the RISAF, write poison bit sent to DDRCTRL
10	um_ddrss_axi1_arpoison and um_ddrss_axi1_arvalid	RISAF	At the output of the RISAF, read poison bit sent to DDRCTRL
11	um_ddrss_axi1_ruser[0] and um_ddrss_axi1_rvalid	DDRCTRL	Read transaction must be decrypted by RISAF
12	um_perf_op_is_enter_selfref	DDRCTRL	Assert for every entry into self-refresh
13	um_lpr_credit_cnt == 0b0 0b1: 0b0	DDRCTRL	Signal asserted when there is no more space in the low priority read cam
14	um_hpr_credit_cnt == 0b0 0b1: 0b0	DDRCTRL	Signal asserted when there is no more space in the high priority read cam
15	um_wr_credit_cnt == 0b0 0b1: 0b0	DDRCTRL	Signal asserted when there is no more space in the write cam

The figure below presents the observability signals sent to HDP from the DDRSS.

Figure 36. Observability infrastructure organization



1. These 16 signals are detailed in *Table 56*.
2. These 12 signals are detailed in *Table 55*.

12.5.5 DDRSS low power

The DDRSS low-power strategy is a complex joint work involving the RCC, the PWR, the DDRCTRL, and the DDRPHYC. At DDRSS level, registers are used to control two low-power strategy features.

DDRCTRL XPI port clock gating

The DDRCTRL XPI ports (AXI ports) can be clocked down if there are no traffic going through for a programmable amount of time. This duration is programmable in DDRCTRL registers. The clock gating logic however is outside of the DDRCTRL. The DDRDBG

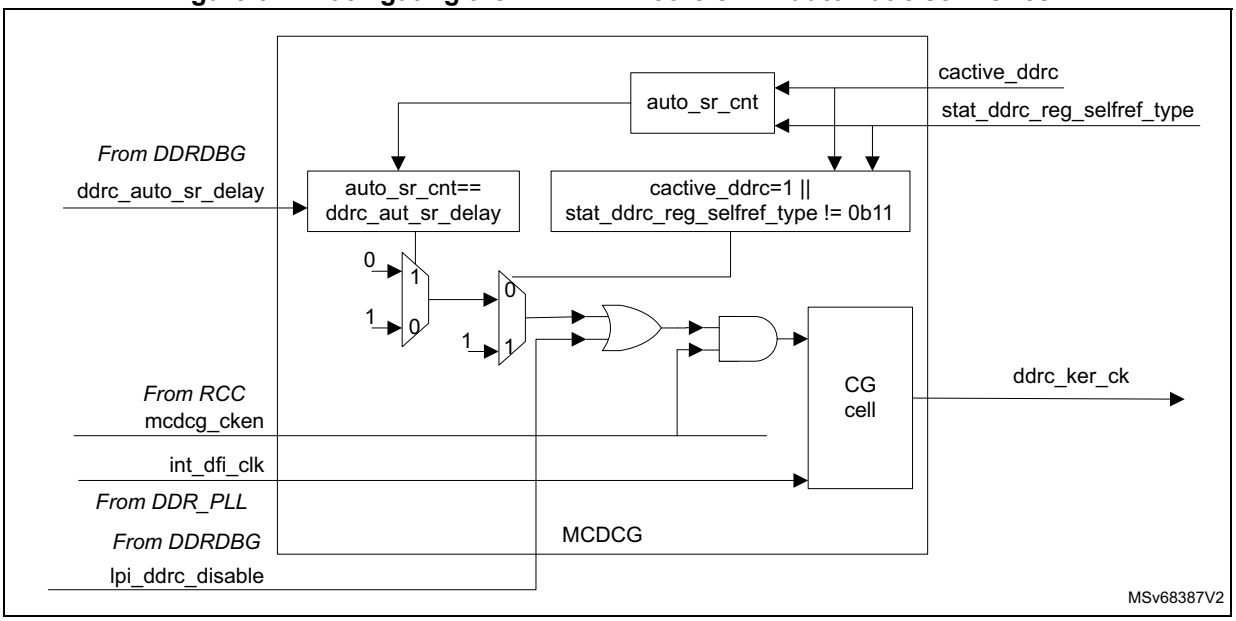
register lp_disable bitfield lpi_xpi_disable can be used to disable this clock gating feature for both xpi ports simultaneously.

Automatic self-refresh involving DDRCTRL clock gating

The STM32MP21xx supports three ways to set the SRAM in self-refresh: automatic, hardware, and software self-refreshs.

An automatic self-refresh relies on DDRCTRL internal counters programmed by software. When the DDRCTRL is idle for an expired counter period, it can set the SDRAM in self-refresh. Once the SDRAM is in self-refresh, the DDRCTRL asserts an external signal (stat_ddrc_reg_selfref_type). A glue logic in the DDRSS relies on this signal to gate down the clock delivered to the DDRCTRL core part, as depicted in the figure below.

Figure 37. Clock gating the DDRCTRL core clk in automatic self-refresh



At DDRSS top level, the registers are used:

- to enable/disable the gating of the clock in automatic self refresh (lp_disable bitfield lpi_ddrc_disable register)
- to control the delay before gating down the clock to the DDRCTRL core part (ddrc_auto_sr_delay bitfield delay_cycles register).

Note: DDRCTRL registers are in the gated-down clock domain. The delay must let any ongoing transaction on the DDRCTRL registers complete.

12.5.6 DFI errors

The DDRPHYC uses the dfi_error signal at the DFI interface to report two kind of errors to the system:

- When the DDRPHYC internal counter DFIPHYUPDCNT expires, the DDRPHYC issues to the DDRCTRL an update request to perform an internal delay element compensation and to avoid that temperature and voltage drifts affect the memory operation. If the DDRCTRL does not acknowledge this request in time, the DDRPHYC

asserts the dfi_error_signal with dfi_error_info = 0b0000. In this case, the DDRPHYC compensation time is over and correct memory operation is no longer guaranteed.

- In LPDDR4, the DDRPHYC must perform periodic DRAM drift compensation operations. The DDRPHYC requires the ownership of the DRAM through dfi_phymstr_req interface, and the DDRCTRL must answer with dfi_phymstr_ack in a given time ($t_{phymstr_resp}$). If the DDRCTRL has not acknowledged the request within this time, a dfi_error is asserted with dfi_error_info = 0b0001.

The dfi_error is reported at the top of the DDRSS as an interrupt, and status is logged in the DDRSS DDRDBG.

12.6 DDRSS registers

12.6.1 DDRDBG low-power disable register (DDRDBG_LP_DISABLE)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPI_DDRC_DISABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPI_XPI_DISABLE
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **LPI_DDRC_DISABLE**:
0: No effect
1: Disable automatic clock gating of DDRC when idle.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LPI_XPI_DISABLE**:
0: No effect
1: Disable automatic clock gating of XPIs when idle.

12.6.2 DDRDBG bypass P_{CLKEN} register (DDRDBG_BYPASS_PCLKEN)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: No effect

1: The DDRPHYC Bypass_Pclken is active. The targeted data rate is less than 666 MT/s. The clock fed to the SDRAM interface is the one coming directly from the interfan frac pll, and the DDRPHYC does not alter it.

12.6.3 DDRDBG DDRC auto SR delay register (DDRDBG_DDRC_AUTO_SR_DELAY)

Address offset: 0x008

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DELAY_CYCLES[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DELAY_CYCLES[7:0]**: Number of delay cycles to wait before to gate down the DDRC clock when in auto self-refresh

This delay allows any ongoing register access to be completed before to gate the DDRC clock.

12.6.4 DDRDBG OBSP control register (DDRDBG_OBSP_CTRL)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	OBSP3_SEL[4:0]					Res.	Res.	Res.	OBSP2_SEL[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OBSP1_SEL[4:0]					Res.	Res.	Res.	OBSP0_SEL[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **OBSP3_SEL[4:0]**:

Selects which of the input observable signal is forwarded to the HDP on selectable observability line 3.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **OBSP2_SEL[4:0]**:

Selects which of the input observable signal is forwarded to the HDP on selectable observability line 2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **OBSP1_SEL[4:0]**:
Selects which of the input observable signal is forwarded to the HDP on selectable observability line 1.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **OBSP0_SEL[4:0]**:
Selects which of the input observable signal is forwarded to the HDP on selectable observability line 0.

12.6.5 DDRDBG FRAC PLL lock register (DDRDBG_FRAC_PLL_LOCK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LOCK**:
0: No effect
1: PLL lock, direct copy of ddrss_pll2_lock

12.6.6 DDRDBG DTO output register (DDRDBG_DTO_OUTPUT)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTO
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DTO**:
0: No effect
1: DDRPHYC DTO output is stored in this register. Among other signals of interests, the DDRPHYC PLL lock can be observed in this register.

12.6.7 DDRDBG DFI error control register (DDRDBG_DFI_ERRORS_CTRL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_SET	DFI_ERROR1_CLEAR	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_SET	DFI_ERROR0_CLEAR
						rw	rw							rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DFI_ERROR1_SET**:
0: No effect
1: Trigger the interrupt linked with dfi_error.

Bit 8 **DFI_ERROR1_CLEAR**:
0: No effect
1: Clear the interrupt linked with dfi_error1.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DFI_ERROR0_SET**:
0: No effect
1: Trigger the interrupt linked with dfi_error0.

Bit 0 **DFI_ERROR0_CLEAR**:
0: No effect
1: Clear the interrupt linked with dfi_error0.

12.6.8 DDRDBG DFI error status register (DDRDBG_DFI_ERROR_STATUS)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_STATUS
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DFI_ERROR1_STATUS**: Status bit for dfi_error1
0: No effect
1: Clear the interrupt linked with dfi_error1.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DFI_ERROR0_STATUS**: Status bit for dfi_error0
0: No effect
1: Clear the interrupt linked with dfi_error0.

12.6.9 DDRDBG DFI error mask register (DDRDBG_DFI_ERROR_MSK)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_MASK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_MASK
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DFI_ERROR1_MASK**:
0: No effect
1: Interrupt linked to dfi_error1 is masked. Clear the interrupt linked with dfi_error1.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DFI_ERROR0_MASK**:
0: No effect
1: Interrupt linked to dfi_error0 is masked. Clear the interrupt linked with dfi_error0.

12.6.10 DDRDBG DDR34 AC swizzle ADD3_0 register (DDRDBG_DDR34_AC_SWIZZLE_ADD3_0)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD3[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD2[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD0[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_ADD3[4:0]**: Remapping input bit on add3 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_ADD2[4:0]**: Remapping input bit on add2 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_ADD1[4:0]**: Remapping input bit on add1 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_ADD0[4:0]**: Remapping input bit on add0 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Table 57. Detailed AC swizzling register values

Register value (binary format)	Selected remapping (selected input signal)	Register value (binary format)	Selected remapping (selected input signal)
00000	No remapping	01110	DFI_ADDRSS_13
00001	DFI_ADDRSS_0	01111	DFI_ADDRSS_14 (DDR3 only)
00010	DFI_ADDRSS_1	10000	DFI_ADDRSS_15 (DDR3 only)
00011	DFI_ADDRSS_2	10001	Reserved
00100	DFI_ADDRSS_3	10010	DFI_ACTN (DDR4 only)
00101	DFI_ADDRSS_4	10011	DFI_BANK_0
00110	DFI_ADDRSS_5	10100	DFI_BANK_1
00111	DFI_ADDRSS_6	10101	DFI_BANK_2
01000	DFI_ADDRSS_7	10110	DFI_BANK_0
01001	DFI_ADDRSS_8	10111	DFI_BANK_1
01010	DFI_ADDRSS_9	11000	DFI_CASN
01011	DFI_ADDRSS_10	11001	DFI_RASN
01100	DFI_ADDRSS_11	11010	DFI_WEN
01101	DFI_ADDRSS_12	11011 to 11111	Reserved

12.6.11 DDRDBG DDR34 AC swizzle ADD7_4 register (DDRDBG_DDR34_AC_SWIZZLE_ADD7_4)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD7[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD6[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD5[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD4[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_ADD7[4:0]**: Remapping input bit on add7 DDRPHYC input at the DFI interface (remapping is defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_ADD6[4:0]**: Remapping input bit on add6 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_ADD5[4:0]**: Remapping input bit on add5 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_ADD4[4:0]**: Remapping input bit on add4 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

12.6.12 DDRDBG DDR34 AC swizzle ADD11_8 register (DDRDBG_DDR34_AC_SWIZZLE_ADD11_8)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD11[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD10[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD9[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD8[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_ADD11[4:0]**: Remapping input bit on add11 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_ADD10[4:0]**: Remapping input bit on add10 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_ADD9[4:0]**: Remapping input bit on add9 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_ADD8[4:0]**: Remapping input bit on add8 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

12.6.13 DDRDBG DDR34 AC swizzle ADD15_12 register (DDRDBG_DDR34_AC_SWIZZLE_ADD15_12)

Address offset: 0x10C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD15[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD14[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD13[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD12[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_ADD15[4:0]**: Remapping input bit on add15 DDRPHYC input at he DFI interface (remapping defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_ADD14[4:0]**: Remapping input bit on add14 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_ADD13[4:0]**: Remapping input bit on add13 DDRPHYC input at he DFI interface (remapping defined in *Table 57*)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_ADD12[4:0]**: Remapping input bit on add12 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

12.6.14 DDRDBG DDR34 AC swizzle BK2 0 ACTN register (DDRDBG_DDR34_AC_SWIZZLE_BK2_0_ACTN)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK3[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK2[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ACTN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_BK3[4:0]**: Remapping input bit on bk3 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_BK2[4:0]**: Remapping input bit on bk2 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_BK1[4:0]**: Remapping input bit on bk1 DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_ACTN[4:0]**: Remapping input bit on actn DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

12.6.15 DDRDBG DDR34 AC swizzle RASN CASN BG1_0 register (DDRDBG_DDR34_AC_SWIZZLE_RASN_CASN_BG1_0)

Address offset: 0x114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DDR34_AC_SWIZZLING_RASN[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_CASN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DDR34_AC_SWIZZLING_BG1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BG0[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DDR34_AC_SWIZZLING_RASN[4:0]**: Remapping input bit on rasn DDRPHYC input at the DFI interface (remapping defined in *Table 57*)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DDR34_AC_SWIZZLING_CASN[4:0]**: Remapping input bit on casn DDRPHYC input at the DFI interface (remapping defined in *Table 57*

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DDR34_AC_SWIZZLING_BG1[4:0]**: Remapping input bit on bg1 DDRPHYC input at the DFI interface (remapping defined in *Table 57*

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_BG0[4:0]**: Remapping input bit on bg0 DDRPHYC input at the DFI interface (remapping defined in *Table 57*

12.6.16 DDRDBG DDR34 AC swizzle WEN register (DDRDBG_DDR34_AC_SWIZZLE_WEN)

Address offset: 0x118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDR34_AC_SWIZZLING_WEN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DDR34_AC_SWIZZLING_WEN[4:0]**: Remapping input bit on wen DDRPHYC input at the DFI interface (remapping defined in *Table 57*

12.6.17 DDRSS register map

Table 58. DDRSS register map and reset values

offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DDRDBG_LP_DISABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPI_DDRC_DISABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPI_XPI_DISABLE
	Reset value																								0								0
0x004	DDRDBG_BYPASS_PCLKEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0x008	DDRDBG_DDRC_ AUTO_SR_DELAY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DELAY_CYCLES[7:0]							
	Reset value																									0	0	1	0	0	0	0	0
0x00C	DDRDBG_OBPS_CTRL	Res.	Res.	Res.	OBSP3_SEL [4:0]					Res.	Res.	Res.	OBSP2_SE L[4:0]					Res.	Res.	Res.	OBSP1_SEL [4:0]					Res.	Res.	Res.	OBSP0_SE L[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x010	DDRDBG_FRAC_PLL_LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x014	DDRDBG_DTO_OUTPUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTO
	Reset value																																0
0x018-0x02C	Reserved	Reserved																															
0x030	DDRDBG_DFI_ ERRORS_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_SET	DFI_ERROR1_CLEAR	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_SET	DFI_ERROR0_CLEAR
	Reset value																							0	0							0	0

Table 58. DDRSS register map and reset values (continued)

offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	DDRDBG_DFI_ ERROR_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_STATUS
	Reset value																								0								0
0x038	DDRDBG_DFI_ ERROR_MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR1_MASK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ERROR0_MASK
	Reset value																								0								0
0x03C - 0x0FC	Reserved	Reserved																															
0x100	DDRDBG_DDR34_AC_ SWIZZLE_ADD3_0	Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD3[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD2[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD0[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	
0x104	DDRDBG_DDR34_AC_ SWIZZLE_ADD7_4	Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD7[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD6[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD5[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD4[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	
0x108	DDRDBG_DDR34_AC_ SWIZZLE_ADD11_8	Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD11[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD10[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD9[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD8[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	
0x10C	DDRDBG_DDR34_AC_ SWIZZLE_ADD15_12	Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD15[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD14[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD13[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ADD12[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	

Table 58. DDRSS register map and reset values (continued)

offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x110	DDRDBG_DDR34_AC_SWIZZLE_BK2_0_ACTN	Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK3[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK2[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BK1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_ACTN[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x114	DDRDBG_DDR34_AC_SWIZZLE_RASN_CASN_BG1_0	Res.	Res.	Res.	DDR34_AC_SWIZZLING_RASN[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_CASN[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BG1[4:0]					Res.	Res.	Res.	DDR34_AC_SWIZZLING_BG0[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x118	DDRDBG_DDR34_AC_SWIZZLE_WEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDR34_AC_SWIZZLING_WEN[4:0]				
	Reset value																												0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

13 DDR3L/DDR4/LPDDR4 controller (DDRCTRL)

13.1 DDRCTRL introduction

The DDRCTRL is a DDR-SDRAM controller. Combined with the DDRPHYC (DDR PHY with PUBL block), it provides a complete memory interface solution for the DDR memory subsystem.

13.2 DDRCTRL features

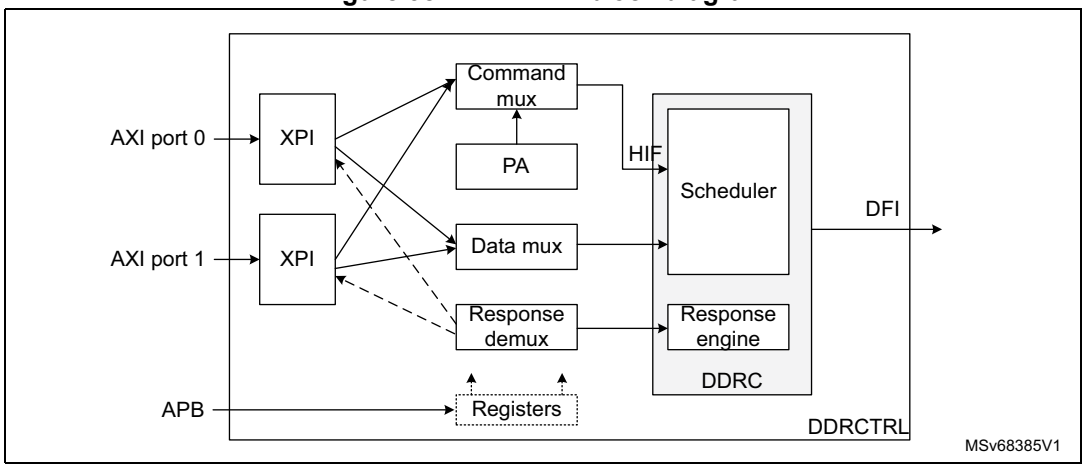
- Two 128-bit AXI port interfaces (XPIs)
- AXI clock asynchronous to the controller
- Supported standards
  - JEDEC compliant LPDDR4 SDRAM up to 1600 MT/s
  - JEDEC compliant DDR4 SDRAM up to 1600 MT/s with DLL on-range
  - JEDEC compliant DDR3L SDRAM up to 1600 MT/s with DLL on-range
  - Single- or dual-rank for LPDDR4 devices
  - Single- or dual-rank for DDR3L/DDR4 devices
- 64 bits with DFI 4.0 compliant interface to DDRPHYC
- Advanced scheduler and SDRAM command generator
- 1:2 frequency ratio between DDRC clock and DDR PHY/external memory clock
- Asynchronous clock between AXI ports and DDRC (clock crossing local to XPIs)
- Programmable full data width (32-bit) or half data width (16-bit)
- Options to avoid starvation of lower-priority traffic
- Guaranteed coherency for write-after-read (WAR) and read-after-write (RAW) on AXI ports
- Programmable support for burst length options (8, 16)
- Write combine (multiple writes to the same address combined into a single write)
- Automatic SDRAM power-down entry/exit caused by a lack of transaction arrival for programmable time
- Automatic clock stop entry/exit caused by lack of transaction arrival
- Automatic low-power mode operation caused by a lack of transaction arrival for programmable time via hardware low-power interface
- Programmable paging policy selectable as any of the following:
  - Leave page open after accesses (open page policy).
  - Close page when no further accesses available in the controller for that page.
  - Auto-precharge with each access, with an optimization for page-close mode which leaves the page open after a flush for read-write and write-read collision cases.
- Self-refresh entry/exit as follows:
  - Automatic self-refresh entry/exit caused by lack of transaction arrival for programmable time
  - Self-refresh entry/exit under software control
- Explicit SDRAM mode register updates under software control

- Flexible address mapper logic to allow application specific mapping of (rank,) row, column, bank bits
- Programmable support for 1T or 2T timing
- User-selectable refresh control options:
  - Controller-generated auto-refreshes at programmable average intervals
  - Ability to group up to eight controller-generated refreshes together to be issued consecutively (this reduces the frequency of page closings, and increases the overall efficiency)
  - When controller-generated refreshes are grouped, some refreshes can be issued speculatively when the controller is idle for a programmable period of time
  - Ability to disable controller-generated auto-refreshes
  - Ability to issue a refresh through direct software request

13.3 DDRCTRL block diagram

A simplified DDRCTRL block diagram is shown in the figure below.

Figure 38. DDRCTRL block diagram



13.4 DDRCTRL architecture overview

The DDRCTRL block diagram in *Figure 38* shows:

- the AXI port interface (XPI): provides the interface to the AXI port with the AXI bus protocol handling, data buffering and reordering for read data, data bus size/clock domain conversion and memory burst address alignment.
- the port arbiter (PA): provides latency sensitive, priority-based arbitration between the addresses issued by the XPIs.
- the DDR controller (DDRC): contains a logical CAM (content addressable memory). This holds information on the commands and is used by the scheduling algorithms to optimally schedule commands to be sent to the PHY, based on priority, bank/rank status and DDR timing constraints.
- the APB register: contains the software accessible registers.

- the write data: stored until its associated command is issued to the PHY. The write data buffer is local to the DDRC.

The read data are handled by the response engine in the DDRC, and returned in the order of scheduled read commands on the HIF (host interface). the read data are stored in the read reorder buffer (local to XPI), and returned in order to the AXI ports.

13.4.1 AXI port interface (XPI)

The XPI processes the AXI information: address, burst size, burst length, and burst type. It converts the AXI burst into read/write requests which are forwarded to the PA.

In the opposite direction, the XPI converts the responses from the DDRC into appropriate AXI responses.

Read address channel features

- Supports any burst length for incremental bursts (≤ 256) and burst with length ≤ 16 for WRAP bursts.
- The start address can be unaligned to the AXI width.
- The burst size can be less than the AXI width.
- The read-address queue (RAQ) is configured with a single queue.
- The clock crossing is performed in RAQ.
- The depth of RAQ is set to four.
- The generation of read request is based on AXI address and burst information, than is divided into packets equal to memory burst length (BL2, BL4, BL8, BL16).
- In case of an unaligned burst, the first read request is unaligned, and the remaining read requests are aligned.

Note: The device interconnect performs the realignment to a memory burst boundary. It may cause some data beats to be discarded (affecting bandwidth), and introduce additional latency on the read data and response channel.

Write address channel features

- The write address queue (WAQ) is used to store all addresses for write requests from a given port.
- There is a single queue for all AXI IDs from a given port.
- The write address channel behavior is similar to the read address channel.
- The clock crossing is performed in WAQ.
- The depth of WAQ is set to four.

Read data and response channels

- The read data and response channel has a single data storage queue (RDQ).
- The clock crossing is performed in RDQ.
- The depth of RDQ is set to 10.
- The data from different IDs are stored in the same queue, and are returned in the order of read address acceptance.
- A read reorder buffer is implemented in each port to reorder the read data for that port to the same order as the order of the AXI read commands.
- The virtual channels and dynamic mapping allow optimum reordering of read data.

13.4.2 Port arbiter (PA)

The PA arbitrates command requests from the two AXI ports to the HIF of the DDR controller. The PA is comprised of multiple tiers of arbitration stages which include:

- read/write arbitration (single HIF option)
- two priority level arbitration, based on port aging and expired-VPR/expired-VPW commands (timeout - priority0)
- two priority level arbitration for read requests based on DDRC read priorities (HPR/LPR-VPR)
- 32 priority level arbitration based on internal port aging, or 16 priority level arbitration based AXI QoS inputs (programmed in AXI interconnect)
- round-robin arbitration to resolve ports having the same priority after passing all stages of arbitration

Note: The port aging refers to counting down the time when a port has a request, but is not granted access to the DDRC.

The port timeout refers to when the port age becomes 0, which is also referred as the highest priority -priority0.

The expired-VPR refers to variable priority read commands whose counter has expired. Read/write credits refer to the DDRC 'Credit Mechanism' employed at the HIF interface.

13.4.3 Host interface (HIF)

The HIF is an internal interface to the DDRCTRL (AXI port interface and PA handle all signals on this interface). It is only described for the sake of understanding QoS and of multi-tier arbitration.

The read/write requests and write data are transferred from PA to DDRC using HIF.

The requests are throttled by:

- a credit mechanism that ensures the buffer space is available for any request made to the DDRC, prior to the request being made
- an independent stall mechanism that throttles requests when address collisions take place, or whether entering a self-refresh via the software self-refresh.

The read data are sent back via a common response interface

Credit mechanism

The DDRC uses a credit mechanism to ensure that buffers do not overflow. The interface making the request to the DDRC, can only request commands for which it has been granted credits to issue.

The credits are tracked separately for the following three command types:

- high-priority read (HPR)
- low-priority read (LPR)
- write (WR)

The interface logic includes counters to track the number of credits. The credit count increments every time the DDRC issues a credit. VPR commands are counted in the LPR credit bucket. VPW commands are counted in the WR credit bucket.

Read requests

When the LPR or HPR credit count is greater than zero, the interface can issue read requests to the DDRC.

Write requests

When the WR credit count is greater than zero, the interface can issue write requests to the DDRC.

13.4.4 DDR scheduler (DDRC)

The DDRC queues commands for scheduling and for the SDRAM command generation unit. The DDRC includes the following stores:

- HPR and LPR stores in common read CAM
- WR store in write CAM

Both CAMs are 16 entries. HPR/LPR split is software programmable.

The DDRC allows the user to manage the TSC (transaction service control) according to QoS and timeouts.

13.5 DDRCTRL functional description

13.5.1 Transaction service control (TSC) and quality of service (QoS)

The TSC allows the user to carefully manage:

- costly read/write bus turn-around
- priorities of read requests to generally favor high-priority traffic, while also preventing starvation of low priority traffic

This functionality is implemented in a simple 2-state machine for each traffic type, with completely configurable controls. The states of the 2-state machine determine when reads/writes are serviced, and the relative priority (high-priority versus low-priority reads) at any time.

The transactions are separated into the following read/write classes:

- low-priority reads (LPR)
- variable-priority reads (VPR)
- high-priority reads (HPR)
- normal-priority writes (NPW)
- variable-priority writes (VPW)

These traffic classes are determined according to AXI QoS values (AxQOS[3:0]) set by the device AXI interconnect (AXIM) and QoS region settings.

HPR and VPR/LPR are stored in a common read CAM which is split according to LPR_NUM_ENTRIES in DDRCTRL_REGS_SCHED.

VPR commands move to the expired-VPR after a timeout set by RQOS_MAP_TIMEOUTB in DDRCTRL_MP_PCFGQOS1_x. The expired-VPR goes to highest priority (Priority0).

VPW and NPW are stored in a common write CAM. VPW commands go to the expired-VPR after timeout set by WQOS_MAP_TIMEOUT2 in DDRCTRL_MP_PCFGWQOS1_x. Expired-VPW commands go to highest priority (Priority0).

The following registers are related to TSC and QoS control:

- DDRCTRL_REGS_SCHED: main scheduler control
- DDRCTRL_REGS_SCHED1: page close timer
- DDRCTRL_REGS_PERFHPR1: controls HPR starvation and run_length
- DDRCTRL_REGS_PERFLPR1: controls LPR starvation and run_length
- DDRCTRL_MP_PCFGR_0/1: AXI port read control for port 0/1
- DDRCTRL_MP_PCFGW_0/1: AXI port write control for port 0/1
- DDRCTRL_MP_PCFGQOS0_x or DDRCTRL_MP_PCFGQOS1_x (where x is the port number): AXI port read QoS region mapping for port 0/1 and port timeouts.
- DDRCTRL_MP_PCFGWQOS0_x or DDRCTRL_MP_PCFGWQOS1_x (where x is the port number): AXI port write QoS region mapping for port 0/1 and port timeouts.

13.5.2 Paging policy options

Open page policy is set by PAGECLOSE = 0 in DDRCTRL_REGS_SCHED. Any bank remains open until there is a need to close it because of a page miss or refresh timeout.

Intelligent precharge is a mid way between open- and close-page policies, set by PAGECLOSE = 1 in DDRCTRL_REGS_SCHED. PAGECLOSE_TIMER value in DDRCTRL_REGS_SCHED1 gives the time waited when there are no page hits to a bank in the CAM, before the auto-precharge or the explicit precharge is scheduled.

This timer allows the page to be kept open for a configurable number of clock cycles after there are no more commands pending in the CAM to a bank. Intelligent precharge is an option between open page and closed page

13.5.3 Power saving features

The DDRCTRL supports various methods to save power within the system using power saving opportunities in the SDRAM, the DDRPHYC, and the device.

Precharge power down (PDN)

The DDRCTRL automatically puts the SDRAM in PDN under the control of following parameters:

- POWERDOWN_EN in DDRCTRL_REGS_PWRCTRL to enter PDN (PDN entry, PDE) after a number of idle cycle
- POWERDOWN_TO_X32 in DDRCTRL_REGS_PWRTMG: number of idle clocks in multiple of 32 to enter PDN
- T_XP in DDRCTRL_REGS_DRAMTMG1: minimum time to wait after PDN exit (PDX) to any command (DLL stays locked during PDN)

Note: During PDN, refresh commands are performed and clocks must be maintained.
The DDRCTRL precharges all banks before PDE.

DDR3L slow power down (with MR0[12] =1) is not supported.

Self-refresh (SR)

The SR is a significant power saving opportunity for the SDRAM. During SR, the DDRPHYC provides additional power savings, and the DDRCTRL/DDRPHYC clocks can be stopped.

The DDRCTRL puts the SDRAM in SR in the following cases:

- automatic SR, when there are no read/write for a predefined time, controlled by:
  - SELFREF_EN in DDRCTRL_REGS_PWRCTRL: to enter SR (SRE) after a number of idle cycle
  - SELFREF_TO_X32 in DDRCTRL_REGS_PWRTMG: number of idle clocks in multiple of 32 to enter SR
- software controlled SR, controlled by SELFREF_SW in DDRCTRL_REGS_PWRCTRL: setting this bit is referred as software SRE.
- hardware controlled SR, supported by the FSM external to the DDRCTRL, and under the RCC control (see *Section 20: Reset and clock control (RCC)* for mode details). When the devices goes in Stop mode, the DDRCTRL can put the SDRAM into SR.

Maximum power saving mode (MPSM)

- MPSM is applicable only to DDR4.
- These power-saving opportunities are supported but do not present any significant interest because the SDRAM content is lost during DPD (Deep power-down) and MPSM, that are practically not different from shutting down the DDR.

DDR3L/DDR4 DLL off mode

The DDR3L and DDR4 DLL off modes are not supported.

13.5.4 Address mapper

The DDRCTRL address mapper converts the AXI port read/write request into row, bank, column (rank if any) SDRAM address at XPI level. The address mapping is very flexible and

defined according to DDRCTRL_REGS_ADDRMAP1 to DDRCTRL_REGS_ADDRMAP11 settings.

13.5.5 DRAM timing parameters

The DDRCTRL schedule commands to the SDRAM according to multiple timing constraints, as programmed in DDRCTRL_REGS_DRAMTMGx, according to SDRAM JEDEC timing parameters versus frequency.

The following registers are related to timings constrains:

- DDRCTRL_REGS_DRAMTMG0 to DDRCTRL_REGS_DRAMTMG17
- DDRCTRL_REGS_ODTTMG
- DDRCTRL_REGS_DFITMG0/1

13.5.6 SDRAM initialization sequence

The initialization is controlled by the DDRPHYC supporting a PUBL built-in initialization sequence for each DDR type, and triggered by PIR registers (see *Section 14: DDR physical interface control (DDRPHYC)* for more details).

13.5.7 Refresh controls

The refresh can be issued using one of the following method:

- using the direct software request of the refresh command (DIS_AUTO_REFRESH = 1 in DDRCTRL_REGS_RFSHCTL3)
- the auto-refresh feature in the DDRCTRL (DIS_AUTO_REFRESH = 0)

Refresh using the direct software request of refresh command

Follow these steps:

1. Set DIS_AUTO_REFRESH = 1. DDRCTRL checks for any pending refreshes. Any pending refreshes are issued right away using the ‘critical refresh’ feature inside the DDRCTRL. After these refreshes are issued, all refresh timers inside the DDRCTRL are reset to 0. They are re-activated only when the auto-refresh feature is enabled.
2. The device core must keep track of the refresh requirements of the SDRAM.
3. The refresh command can be issued by setting RANK0_REFRESH to 1 in DDRCTRL_REGS_DBGCMD. When the refresh request is stored, the corresponding bit is automatically cleared. The device core can initiate a refresh operation only if RANK0_REFRESH _BUSY = 0 in DDRCTRL_REGS_DBGSTAT. The DDRCTRL issues refresh to the SDRAM at the earliest.
4. Software-driven refresh commands are loaded into a 9-entry buffer, and are issued by the DDRCTRL on the DFI as soon as it is legal to do so. If the buffer saturates, RANK0_REFRESH _BUSY remains asserted to prevent software from initiating further refreshes.

Refresh using the auto-refresh feature inside DDRCTRL

The DDRCTRL provides advanced refresh controls. Besides fully-configurable refresh constraints ($t_{RFC(min)}$ and t_{REFI}), it can also be programmed to gather refreshes to each rank of the SDRAM, to reduce the bandwidth consumed by refreshes, and to increase the

likelihood that refreshes can be serviced during an idle period. The following registers are related to refresh controls:

- DDRCTRL_REGS_RFSHCTL0
- DDRCTRL_REGS_RFSHCTL3
- DDRCTRL_REGS_RFSHTMG

The purposes of refresh control are the following:

- reduce the bandwidth impact of refresh cycles
- increase the likelihood of refreshes being serviced during idle periods
- provide fine-gain control of the trading-off between above benefits (to gather refreshes) versus the increased worst-case latencies associated with gathering refreshes

For more information, refer to each register description. Beside issuing refresh at regular intervals according to $t_{RFC(min)}$ and t_{REFI}, several optimization options are possible: burst refresh, speculative refresh, and per-bank refresh (applicable to LPDDR4).

13.5.8 ZQ calibration

This feature is applicable to all DDR types. The DDRCTRL can use the ZQ calibration command to calibrate SDRAM RON (resistor on) and ODT (on-die termination) values over PVT (process, voltage, temperature).

The DDRPHYC supports RON and ODT calibration at initialization (see *Section 14: DDR physical interface control (DDRPHYC)* for more details).

The DDRCTRL can issue periodic calibrations which take less time at regular interval (ZQCS). The DDRCTRL can also issue a long calibration (ZQCL) after SR exit.

The following registers are related to ZQ calibration controls:

- DDRCTRL_REGS_ZQCTL0/1/2
- DDRCTRL_REGS_ZQSTAT

For more information on ZQ Calibration, refer to JEDEC specifications of the corresponding SDRAM.

13.5.9 Register programming

Most registers are static and must be programmed only when DDRCTRL is under reset.

Dynamic registers can be programmed at any time.

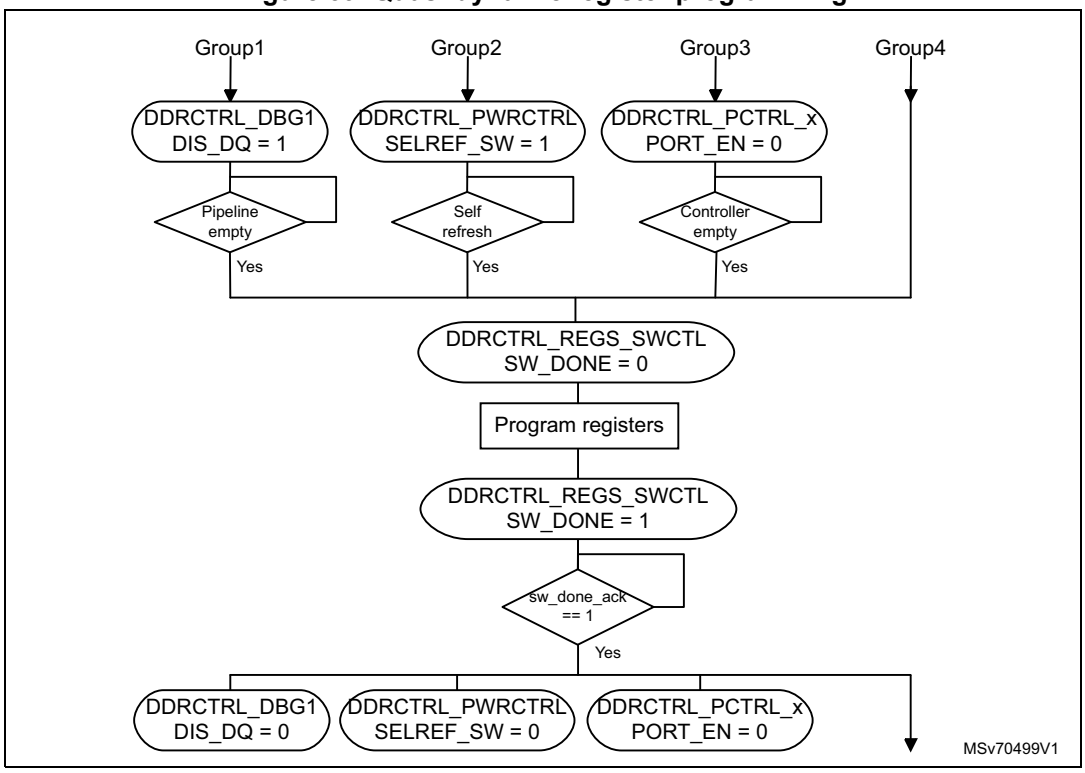
Refresh related registers are dynamic registers but the following steps are needed to update them:

- Change the refresh-related parameter as desired.
- After the changed register is stable, toggle REFRESH_UPDATE_LEVEL in DDRCTRL_REGS_RFSHCTL3.

Quasi-dynamic parameters can be programmed following the flow described in *Figure 39*, with the group as listed in the bitfield programming mode:

- Group 1: parameter that can be written when no read/write traffic is present at DFI (pipeline empty)
- Group 2: parameter that can be written in self-refresh, DPD, or MPSM
- Group 3: parameter that can be written when the controller is empty
- Group 4: parameter that can be written depending on MSTR frequency_mode and MSTR2 target_frequency

Figure 39. Quasi-dynamic register programming



13.6 DDRCTRL configuration

The DDRCTRL is configured with these major hardware options:

- DDR mode support: LPDDR4/DDR4/DDR3L
- frequency ratio: 1:2
- arbiter: include PA with two AXI ports
- CAM depth: 32 with bypass enabled
- single HIF
- single RAQ per port

13.7 DDRCTRL registers

The following variables are considered with fixed values:

- MEMC_NO_OF_ENTRY = 32

13.7.1 DDRCTRL master0 register (DDRCTRL_REGS_MSTR)

Address offset: 0x000

Reset value: 0x0304 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DEVICE_CONFIG[1:0]		Res.	Res.	Res.	Res.	ACTIVE_RANKS[1:0]		Res.	Res.	Res.	Res.	BURST_RDWR[3:0]			
rw	rw					rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLL_OFF_MODE	Res.	DATA_BUS_WIDTH[1:0]		Res.	EN_2T_TIMING_MODE	BURSTCHOP	Res.	Res.	Res.	LPDDR4	DDR4	Res.	Res.	Res.	DDR3L
rw		rw	rw		rw	rw				rw	rw				rw

Bits 31:30 **DEVICE_CONFIG[1:0]**: Device configuration
00: x4 device
01: x8 device
10: x16 device
11: x32 device
Note: Programming mode: Static.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **ACTIVE_RANKS[1:0]**: Only present for multi-rank configurations
Each bit represents one rank. For two-rank configurations, only bits[25:24] are present.
1: Populated
0: Unpopulated
LSB is the lowest rank number. For two ranks, the following combinations are legal:
01: One rank
11: Two ranks
Others: reserved
Note: Programming mode: Static.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **BURST_RDWR[3:0]**: SDRAM burst length
This bitfield controls the burst size used to access the SDRAM. This must match the burst length mode register setting in the SDRAM. For DDR3L, DDR4, this must be set to 0x0100 (BL8).
For LPDDR4, this must be set to 0x1000 (BL16).
0010: Burst length of 4
0100: Burst length of 8
1000: Burst length of 16 (only supported for LPDDR4)
Others: reserved
Note: Programming mode: Static.

Bit 15 **DLL_OFF_MODE**:
1: When the DDRCTRL and DRAM has to be put in DLL-off mode for low frequency operation
0: To put DDRCTRL and DRAM in DLL-on mode for normal frequency operation
Note: Programming mode: Quasi-dynamic Group 2.

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **DATA_BUS_WIDTH[1:0]**: Proportion of DQ bus width that is used by the SDRAM
00: 32 bits
01: 16 bits
10: 8 bits (supported for DDR3L half populated)
Note: Programming mode: Static.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **EN_2T_TIMING_MODE**:
0: DDRCTRL uses 1T timing.
1: DDRCTRL uses 2T timing.
In 2T timing, all command signals (except chip select) are held for two clocks on the SDRAM bus. Chip select is asserted on the second cycle of the command.
2T timing is not supported in LPDDR4 mode and in shared-AC dual channel mode (register value is don't care).
Note: Programming mode: Static.

Bit 9 **BURSTCHOP**: Burst chop (BC4 or 8 on-the-fly) enable in DDR3L/DDR4
Burst chop for reads is exercised only:
- in HIF configurations (DDRCTRL_REGS_INCL_ARB not set)
- in full bus width mode (DATA_BUS_WIDTH = 00 in this register)

BC4 (fixed) mode is not supported.
Note: Programming mode: Static.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPDDR4**: Selects LPDDR4 SDRAM
1: LPDDR4 SDRAM device
0: Non-LPDDR4 device
Note: Programming mode: Static.

Bit 4 **DDR4**: Selects DDR4 SDRAM
1: DDR4 SDRAM device
0: Non-DDR4 device
Note: Programming mode: Static.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DDR3L**: Selects DDR3L SDRAM
1: DDR3L SDRAM device
0: Non-DDR3L SDRAM device
Note: Programming mode: Static.

13.7.2 DDRCTRL operating mode status register (DDRCTRL_REGS_STAT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SELFREF_CAM_NOT_EMPTY	Res.	Res.	SELFREF_STATE[1:0]		Res.	Res.	SELFREF_TYPE[1:0]		Res.	OPERATING_MODE[2:0]		
			r			r	r			r	r		r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SELFREF_CAM_NOT_EMPTY**: Self-refresh (SR) with CAMs not empty

Set to 1 when SR is entered but CAMs are not drained. Cleared after exiting SR.

Note: Programming mode: Static.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **SELFREF_STATE[1:0]**: SR or SRPD (SR power-down) for LPDDR4

This bitfield is used for frequency change and MRR/MRW access during SR.

00: SDRAM not in SR

01: SR 1

10: SRPD

11: SR 2

Note: Programming mode: Static.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SELFREF_TYPE[1:0]**: Flags if SR (except LPDDR4), or SRPD (LPDDR4) is entered, and if it is under automatic SR control only or not

00: SDRAM is not in SR (except LPDDR4) or SRPD (LPDDR4).

11: SDRAM is in SR (except LPDDR4) or SRPD(LPDDR4), which was caused by automatic SR only. If retry is enabled, this ensures that the SR entry (SRE) command is executed correctly without parity error.

10: SDRAM is in SR (except LPDDR4) or SRPD (LPDDR4), which was not caused solely under automatic SR control. It may have been caused by hardware low-power interface and/or software (SELFREF_SW in DDRCTRL_REGS_PWRCTL). If retry is enabled, this ensures that the SRE command is executed correctly without parity error.

01: SDRAM is in SR, which is caused by PHY master request.

Note: Programming mode: Static.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **OPERATING_MODE[2:0]**:
DDR3L design:
00: Init
01: Normal
10: PDN
11: SR
DDR4 design:
000: Init
001: Normal
010: PDN
011: SR
Others: reserved
LPDDR4 design:
000: Init
001: Normal
010: PDN
011: SR/SRSRPD
Others: reserved

13.7.3 DDRCTRL mode read/write control 0 register (DDRCTRL_REGS_MRCTRL0)

Address offset: 0x010

Reset value: 0x0000 0030

Note: Do not enable more than one of the following bitfields simultaneously: SW_INIT_INT, PDA_EN, and MPR_EN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR_WR	PBA_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR_ADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	MR_RANK[1:0]		SW_INIT_INT	PDA_EN	MPR_EN	MR_TYPE
r	r	r	r							rw	rw	rw	rw	rw	rw

Bit 31 **MR_WR**: Mode register (MR) read or write
Setting this bit to 1 triggers a mode register read or write operation.
When the MR operation is complete, the DDRCTRL automatically clears this bit.
The other bitfields of this register must be written in a separate APB transaction, before setting this MR_WR bit.
It is recommended NOT to set this signal if in init operating mode.
Note: Programming mode: Dynamic.

Bit 30 **PBA_MODE**: PBA access execution

When setting this bit to 1 along with setting PDA_EN to 1, the DDRCTRL initiates PBA (per buffer addressability) access instead of PDA (per DRAM addressability) access.
The completion of PBA access is confirmed by PODA_DONE in DDRCTRL_REGS_MRSTAT in the same way as PDA.
0: PDA
1: PBA

Note: Programming mode: Dynamic.

Bits 29:16 Reserved, must be kept at reset value.

Bits 15:12 **MR_ADDR[3:0]**: MR address to be written to

0000: MR0
0001: MR1
0010: MR2
0011: MR3
0100: MR4
0101: MR5
0110: MR6
0111: MR7
Don't care for LPDDR4 (see MR_DATA in DDRCTRL_REGS_MRCTRL1 for mode register addressing in LPDDR4).

Note: Programming mode: Dynamic.

Bits 11:6 Reserved, must be kept at reset value.

Bits 5:4 **MR_RANK[1:0]**: Rank accessed by MR_WR in this register

For multi-rank UDIMMs/RDIMMs/LRDIMMs, which implement address mirroring, it may be necessary to access ranks individually.
Examples (assuming DDRCTRL is configured for four ranks):
0x1: Select rank 0 only
0x2: Select rank 1 only
0x5: Select ranks 0 and 2
0xA: Select ranks 1 and 3
0xF: Select ranks 0, 1, 2, and 3

Note: Programming mode: Dynamic.

Bit 3 **SW_INIT_INT**: Software intervention enable

This bit indicates whether a software intervention is allowed through DDRCTRL_REGS_MRCTRL0/1 before automatic SDRAM initialization routine or not.
For DDR4, this bit can be used to initialize the DDR4 RCD (MR7) before automatic SDRAM initialization.
For LPDDR4, this bit can be used to program additional mode registers before automatic SDRAM initialization if necessary. In LPDDR4 dual channel mode, this bit must be programmed to both channels beforehand.
This bit must be cleared to 0 after completing software operation. Otherwise, the SDRAM initialization routine does not restart.
0: Software intervention not allowed
1: Software intervention allowed

Note: Programming mode: Dynamic.

Bit 2 **PDA_EN**: Mode register operation is MRS in PDA mode or not
0: MRS
1: MRS in PDA
When PBA_MODE = 1 in this register, PBA access is initiated instead of PDA access.
Note: Programming mode: Dynamic.

Bit 1 **MPR_EN**: Mode register operation is MRS or WR/RD for MPR (only supported for DDR4)
0: MRS
1: WR/RD for MPR
Note: Programming mode: Dynamic.

Bit 0 **MR_TYPE**: Mode register operation is read or write (only for LPDDR4/DDR4)
0: Write
1: Read
Note: Programming mode: Dynamic.

13.7.4 DDRCTRL mode read/write control 1 register (DDRCTRL_REGS_MRCTRL1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_DATA[17:16]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR_DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **MR_DATA[17:0]**: Mode register write data for all non-LPDDR4 modes
This bitfield is 18-bit wide in configurations with DDR4 support, and 16-bit in all other configurations.
For LPDDR4, MRCTRL1[15:0] are interpreted as:
- [15:8]: MR address
- [7:0]: MR data for writes, don't care for reads

Note: Programming mode: Dynamic.

13.7.5 DDRCTRL mode register read/write status register (DDRCTRL_REGS_MRSTAT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDA_DONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_WR_BUSY
							r								r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PDA_DONE**: Initiate an MR write operation in PDA/PBA mode only if this bit is low
This bit goes high when three consecutive MRS commands related to the PDA/PBA mode are issued to the SDRAM.
This bit goes low when PDA_EN becomes 0 in DDRCTRL_REGS_MRCTRL0.
It is recommended to write o to PDA_EN after PDA_DONE bit goes high in order to prepare to perform PDA operation next time.
0: Mode register write operation related to PDA/PBA is in progress or has not started yet.
1: Mode register write operation related to PDA/PBA has competed.
Note: Programming mode: Dynamic.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **MR_WR_BUSY**: Initiate an MR write operation only if this bit is low
This bit goes high in the clock after the DDRCTRL accepts the MRW/MRR request.
This bit is low when the MRW/MRR command is issued to the SDRAM.
It is recommended not to perform MRW/MRR commands when this bit is high.
0: The device can initiate a mode register write operation.
1: Mode register write operation is in progress.
Note: Programming mode: Dynamic.

13.7.6 DDRCTRL mode register read/write control 2 register (DDRCTRL_REGS_MRCTRL2)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR_DEVICE_SEL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR_DEVICE_SEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MR_DEVICE_SEL[31:0]**: Device selection during the MRS that happens in PDA mode

Each bit is associated with one device (bit[0] corresponds to device 0, bit[1] to device 1 and so on). For each bit:

1: MRS command must be applied to that device.

0: MRS commands must be skipped for that device.

Note: Programming mode: Dynamic.

13.7.7 DDRCTRL temperature derate enable register (DDRCTRL_REGS_DERATEEN)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DERATE_MR4_PAUSE_FC	DERATE_MR4_TUF_DIS	Res.	RC_DERATE_VALUE[2:0]			DERATE_BYTE[3:0]				Res.	DERATE_VALUE[1:0]		DERATE_ENABLE
		rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DERATE_MR4_PAUSE_FC**: Pauses MR4 reads

Must only be set to 1 as part of frequency change procedures, if DERATE_ENABLE = 1 in this register.

Note: Programming mode: Dynamic.

Bit 12 **DERATE_MR4_TUF_DIS**: Disables use of MR4 TUF flag (MR4[7])

For LPDDR4, it is recommended to set this bit to 1. This affects both the periodic derate logic (DERATE_ENABLE) and DDRCTRL_REGS_DERATE_TEMP_LIMIT_INTR in DDRCTRL_REGS_DERATESTAT.

0: Use MR4 TUF flag (MR4[7]).

1: Do not use MR4 TUF Flag (MR4[7]).

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **RC_DERATE_VALUE[2:0]**: Derate value of t_{RC} for LPDDR4

The required cycles for derating can be determined by dividing 3.75 ns by the core_ddrc_core_clk period, and rounding it up to the next integer.

This bitfield must be set to the previous value - 1.

000: Derating uses +1.

001: Derating uses +2.

010: Derating uses +3.

011: Derating uses +4.

100: Derating uses +5.

101: Derating uses +6.

110: Derating uses +7.

111: Derating uses +8.

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:4 **DERATE_BYTE[3:0]**: MRR data used for derating
Note: Programming mode: Static.

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **DERATE_VALUE[1:0]**: Derate value
The required cycles for derating can be determined by dividing 1.875 ns by the core_ddrc_core_clk period, and rounding it up to the next integer.
This bitfield must be set to the previous value - 1.
00: Derating uses +1.
01: Derating uses +2.
10: Derating uses +3.
11: Derating uses +4.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 0 **DERATE_ENABLE**: Enables derating
This bit must be set to 0 for non-LPDDR4 mode.
0: Timing parameter derating is disabled.
1: Timing parameter derating is enabled using MR4 read value.
Note: Programming mode: Dynamic.

13.7.8 DDRCTRL temperature derate interval register (DDRCTRL_REGS_DERATEINT)

Address offset: 0x024

Reset value: 0x0080 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR4_READ_INTERVAL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR4_READ_INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MR4_READ_INTERVAL[31:0]**: Interval between two MR4 reads
This bitfield value is used to derate the timing parameters.
This register must not be set to zero. Unit: DFI clock cycles
Note: Programming mode: Static.

13.7.9 DDRCTRL temperature derate control register (DDRCTRL_REGS_DERATECTL)

Address offset: 0x02C

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_TEMP_LIMIT_INTR_FORCE	DERATE_TEMP_LIMIT_INTR_CLR	DERATE_TEMP_LIMIT_INTR_EN
													rc_w1	rc_w1	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DERATE_TEMP_LIMIT_INTR_FORCE**: Interrupt force bit for DERATE_TEMP_LIMIT_INTR
Setting this bit to 1 causes the DERATE_TEMP_LIMIT_INTR output pin to be asserted.
At the end of the interrupt force operation, the DDRCTRL automatically clears this bit.
Note: Programming mode: Dynamic.

Bit 1 **DERATE_TEMP_LIMIT_INTR_CLR**: Interrupt clear bit for DERATE_TEMP_LIMIT_INTR
At the end of the interrupt clear operation, the DDRCTRL automatically clears this bit.
Note: Programming mode: Dynamic.

Bit 0 **DERATE_TEMP_LIMIT_INTR_EN**: Interrupt enable bit for DERATE_TEMP_LIMIT_INTR
1: Enabled
0: Disabled
Note: Programming mode: Dynamic.

13.7.10 DDRCTRL low-power control register (DDRCTRL_REGS_PWRCTL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDDR4_SR_ALLOWED	DIS_CAM_DRAIN_SELFREF	STAY_IN_SELFREF	SELFREF_SW	Res.	EN_DFI_DRAM_CLK_DISABLE	Res.	POWERDOWN_EN	SELFREF_EN
							rw	rw	rw	rw		rw		rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **LPDDR4_SR_ALLOWED**: Transition from SRPD to SR, and back to SRPD allowed
This bit cannot be modified while SELFREF_SW = 1 in this register.
0: SRPD -> SR -> SRPD not allowed
1: SRPD -> SR -> SRPD allowed
Note: Programming mode: Dynamic.

Bit 7 **DIS_CAM_DRAIN_SELFREF**: Skipping CAM draining allowed when entering SR
This bit cannot be modified while SELFREF_SW = 1 in this register.
0: CAMs must be empty before entering SR.
1: CAMs are not empty before entering SR (not supported).
Note: Programming mode: Dynamic.

Bit 6 **STAY_IN_SELFREF**: SR state transition
SR is an intermediate state to enter/exit SRPD state for LPDDR4. This bit controls transition from SR state.
1: Transition from SR not allowed
0: Transition from SR allowed
Note: Programming mode: Dynamic.

Bit 5 **SELFREF_SW**: SR state entry
A value of 1 to this bit causes the system to move to SR state immediately, as long as it is not in init operating mode. This is referred to as software entry/exit to SR.
1: Software entry to SR
0: Software exit from SR
Note: Programming mode: Dynamic.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **EN_DFI_DRAM_CLK_DISABLE**: dfi_dram_clk_disable enable
Enables the assertion of dfi_dram_clk_disable whenever a clock is not required by the SDRAM.
0: dfi_dram_clk_disable is never asserted.
1: Assertion of dfi_dram_clk_disable as follows: In DDR3L and DDR4, can only be asserted in SR. In LPDDR4, can be asserted in SRPD, in PDN, and during normal operation (clock Stop).
Note: Programming mode: Dynamic.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **POWERDOWN_EN**: PDN enable
If this bit is set to 1, the DDRCTRL goes into PDN after a programmable number of cycles: maximum idle clocks before power down = POWERDOWN_TO_X32 in DDRCTRL_REGS_PWRTMG.
This bit can be reprogrammed during the course of normal operation.
Note: Programming mode: Dynamic.

Bit 0 **SELFREF_EN**: SDRAM into SR enable
If this bit is set to 1, the DDRCTRL puts the SDRAM into SR after a programmable number of cycles: maximum idle clocks before SR = PWRTMG.SELFREF_TO_X32 in DDRCTRL_REGS.
This bit can be reprogrammed during the course of normal operation.
Note: Programming mode: Dynamic.

13.7.11 DDRCTRL low-power timing register (DDRCTRL_REGS_PWRTMG)

Address offset: 0x034

Reset value: 0x0040 2010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TO_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POWERDOWN_TO_X32[4:0]				
											rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SELFREF_TO_X32[7:0]**: Maximum idle clocks before SR

After this number of clocks of the DDRC command channel being idle, the DDRCTRL automatically puts the SDRAM into SR.

The DDRC command channel is considered idle when there are no HIF commands outstanding. This must be enabled by PWRCTL.SELFREF_EN in DDRCTRL_REGS.

Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **POWERDOWN_TO_X32[4:0]**: Maximum idle clocks before PDN

After this number of clocks of the DDRC command channel being idle, the DDRCTRL automatically puts the SDRAM into PDN.

The DDRC command channel is considered idle when there are no HIF commands outstanding. This must be enabled by POWERDOWN_EN in DDRCTRL_REGS_PWRCTL.

Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

13.7.12 DDRCTRL hardware low-power control register (DDRCTRL_REGS_HWLPCTL)

Address offset: 0x038

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HW_LP_IDLE_X32[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LP_EXIT_IDLE_EN	HW_LP_EN
														rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HW_LP_IDLE_X32[11:0]**: Hardware idle period

The CACTIVE_DDRC output is driven low if the DDRC command channel is idle for the number of cycles defined in this bitfield, if not in init operating mode.

The DDRC command channel is considered idle when there are no HIF commands outstanding.

The hardware idle function is disabled when this bitfield = 0 (1 is an illegal value).

Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Static.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **HW_LP_EXIT_IDLE_EN**: CACTIVE_IN_DDRC pin enable

When this bit is set to 1, the CACTIVE_IN_DDRC pin of the DDRC can be used to exit from the automatic clock stop, automatic power down, or automatic SR.

This does not cause exit of SR that was caused by hardware low-power interface, and/or software (see SELREF_SW in DDRCTRL_REGS_PWRCTL).

Note: Programming mode: Static.

Bit 0 **HW_LP_EN**: Hardware low-power interface enable

Note: Programming mode: Quasi-dynamic Group 2.

13.7.13 DDRCTRL refresh control 0 register (DDRCTRL_REGS_RFSHCTL0)

Address offset: 0x050

Reset value: 0x8021 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFRESH_TO_AB_X32[4:0]					Res.	Res.	Res.	REFRESH_MARGIN[3:0]				Res.	Res.	Res.	REFRESH_TO_X1_X32[4]
rw	rw	rw	rw	rw				rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH_TO_X1_X32[3:0]				Res.	Res.	REFRESH_BURST[5:0]						Res.	PER_BANK_REFRESH	AUTO_REFAB_EN[1:0]	
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:27 **REFRESH_TO_AB_X32[4:0]**: When the DDRCTRL switches automatically from per-bank to all-bank refresh (if enabled by AUTO_REFAB_EN[1:0]), it will use this register to determine when to perform speculative all-bank refreshes.

If the refresh timer (t_{RFCnom}, also known as t_{REFI}) has expired at least once, then a speculative refresh may be performed.

A speculative refresh is a refresh performed at a time when refresh would be useful. When the SDRAM bus is idle for a period of time determined by REFRESH_TO_AB_X32[4:0] and the refresh timer has expired at least once since the last refresh, then a speculative refresh is performed.

Speculative refreshes continues successively until there are no refreshes pending or until new reads or writes are issued to the DDRCTRL.

FOR PERFORMANCE ONLY.

Unit: multiples of 32 DFI clock cycles.

Note: Programming mode: Dynamic - Refresh related.

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:20 **REFRESH_MARGIN[3:0]**: DFI clock cycles before the critical refresh or page timer expires

A critical refresh must be issued before this threshold is reached. It is recommended not to change from the default value 0x2. It must always be less than internally used t_rfc_nom/32n that is equal to:

- T_RFC_NOM_X1_X32 * 32 in DDRCTRL_REGS_RFSHTMG if T_RFC_NOM_X1_SEL = 0 in the same register
- T_RFC_NOM_X1_X32 if T_RFC_NOM_X1_SEL = 1 (for LPDDR4 per-bank refresh only)
- In LPDDR4, t_rfc_nom can be divided by four if derating is enabled by DERATE_ENABLE = 1 in DDRCTRL_REGS_DERATEEN.

Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Dynamic - Refresh related.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:12 **REFRESH_TO_X1_X32[4:0]**:

If the refresh timer (t_{RFCnom}, also known as t_{REFI}) has expired at least once, a speculative refresh can be performed.

A speculative refresh is a refresh performed at a time when refresh is useful. When the SDRAM bus is idle for a period of time determined by this field, and the refresh timer has expired at least once since the last refresh, then a speculative refresh is performed.

Speculative refreshes continues successively until there are no refreshes pending or until new reads or writes are issued to the DDRCTRL.

Unit: DFI clock cycles or multiples of 32 DFI clock cycles, depending on T_RFC_NOM_X1_SEL in DDRCTRL_REGS_RFSHTMG.

Note: Programming mode: Dynamic - Refresh related.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:4 **REFRESH_BURST[5:0]**:

This bitfield + 1 is the number of refresh timeouts that can be accumulated before traffic is blocked, and refreshes are forced to execute.

Closing pages to perform a refresh is a one-time penalty that must be paid for each group of refreshes. Performing refreshes in a burst reduces then the per-refresh penalty of these page closings.

Higher numbers for this bitfield slightly increase utilization. Lower numbers decrease the worst-case latency associated with refreshes.

00000: Single refresh

00001: Burst-of-2 refresh

...

00111: Burst-of-8 refresh

- For more information on burst refresh feature, see section 3.9 of DDR2 JEDEC specification (JESD79-2F.pdf).
- For DDR3L, the refresh is always per-rank and not per-bank. The rank refresh can be accumulated over 8 * t_{REFI} cycles using the burst refresh feature.
- In DDR4 mode, according to the fine-granularity feature, eight refreshes can be postponed in 1X mode, 16 refreshes in 2X mode, and 32 refreshes in 4X mode.
- If using PHY-initiated updates, care must be taken in the setting of this bitfield to ensure that t_{RFCmax} is not violated due to a PHY-initiated update occurring shortly before a refresh burst is due. In this situation, the refresh burst is delayed until the PHY-initiated update is complete.
- In per-bank refresh mode of LPDDR4 (PER_BANK_REFRESH = 1 in this register), 64 refreshes can be postponed.

Note: Programming mode: Dynamic - Refresh related.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PER_BANK_REFRESH**:

Per bank refresh allows traffic to flow to other banks (supported by all LPDDR4 devices).

1: Per bank refresh

0: All bank refresh

Note: Programming mode: Static.

Bits 1:0 **AUTO_REFAB_EN[1:0]**: Enables automatic switching from per-bank to all-bank refresh when the derated refresh period is small.

Automatically switches back to per-bank refresh when the derated refresh period is no longer small.

This register should be programmed to 00 if PER_BANK_REFRESH = 0.

This register should be programmed to 00 for LPDDR4 2Gb and 4Gb.

0x0: Disable automatic switching

0x3: Reserved

0x1: Enable automatic switching when the derated refresh period is $t_{REFI}/2$ or lower

0x2: Enable automatic switching when the derated refresh period is $t_{REFI}/4$

Note: Programming mode: Static.

13.7.14 DDRCTRL refresh control 1 register (DDRCTRL_REGS_RFSHCTL1)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	REFRESH_TIMER1_START_VALUE_X32[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	REFRESH_TIMER0_START_VALUE_X32[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **REFRESH_TIMER1_START_VALUE_X32[11:0]**: Refresh timer start for rank 1 (only present in multirank configurations)

This is useful in staggering the refreshes to multiple ranks to help traffic to proceed.

Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Dynamic - Refresh related.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **REFRESH_TIMER0_START_VALUE_X32[11:0]**: Refresh timer start for rank 0 (only present in multirank configurations)

This is useful in staggering the refreshes to multiple ranks to help traffic to proceed.

Unit: multiples of 32 DFI clock cycles.

Note: Programming mode: Dynamic - Refresh related.

13.7.15 DDRCTRL refresh control 3 register (DDRCTRL_REGS_RFSHCTL3)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_MODE[2:0]			Res.	Res.	REFRESH_UPDATE_LEVEL	DIS_AUTO_REFRESH
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **REFRESH_MODE[2:0]**: Fine granularity refresh mode

This bitfield must be set up while the controller is in reset, or while the controller is in SR mode. Changing this during normal operation is not allowed.

Only fixed 1x mode is supported if DIS_AUTO_REFRESH = 1 in this register.

This bitfield has effect only if a DDR4 SDRAM device is in use (DDR4 = 1 in DDRCTRL_REGS_MSTR).

000: Fixed 1x (normal mode)

001: Fixed 2x

010: Fixed 4x

101: Enable on-the-fly 2x (not supported)

110: Enable on-the-fly 4x (not supported)

Others: reserved

Note: Programming mode: Quasi-dynamic Group 2.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **REFRESH_UPDATE_LEVEL**: Refresh register update

Toggle this bit (either from 0 to 1 or from 1 to 0) to indicate that the refresh registers have been updated.

This bit must not be toggled when the DDRC is in reset (core_ddrc_rstn = 0).

The refresh registers are automatically updated when exiting reset.

Note: Programming mode: Dynamic.

Bit 0 **DIS_AUTO_REFRESH**: Auto-refresh disable

When this bit is 1, the auto-refresh generated by the DDRCTRL is disabled. In this case, the device must generate refreshes using RANKx_REFRESH in DDRCTRL_REGS_DBGCMD.

When this bit transitions from 0 to 1, any pending refreshes are immediately scheduled by the DDRCTRL. For DDR4 only, if the FGR mode is enabled (REFRESH_MODE > 0 in DDRCTRL_REGS_RFSHCTL3), DISABLE AUTO-REFRESH is not supported, and this bit must be set to 0.

This bit can be changed on-the-fly.

Note: Programming mode: Dynamic - Refresh related.

13.7.16 DDRCTRL refresh timing register (DDRCTRL_REGS_RFSHTMG)

Address offset: 0x064

Reset value: 0x0062 008C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
T_RFC_NOM_X1_SEL	Res.	Res.	Res.	T_RFC_NOM_X1_X32[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	T_RFC_MIN[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **T_RFC_NOM_X1_SEL**:
This bit specifies whether values of T_RFC_NOM_X1_X32 in this register and REFRESH_TO_X1_X32 in DDRCTRL_REGS_RFSHCTL0 are x1 or x32.
This applies only when per-bank refresh is enabled (PER_BANK_REFRESH = 1 in DDRCTRL_REGS_RFSHCTL0). Otherwise, x32 register values are used, and this bit is ignored.
0: x32 register values
1: x1 register values
Note: Programming mode: Dynamic - Refresh related.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **T_RFC_NOM_X1_X32[11:0]**: Average time interval between refreshes per rank (7.8 µs for DDR3L and DDR4, see JEDEC specification for LPDDR4)
If DDRCTRL operates in 1:2 mode, set this bitfield to RoundDown(RoundDown(t_{REFI} / t_{CK}) / 2).
In previous cases, if T_RFC_NOM_X1_SEL = 0 in this register, divide the previous result by 32 and round down.

For LPDDR4:
- If using all-bank refreshes (PER_BANK_REFRESH = 0 in DDRCTRL_REGS_RFSHCTL0), use t_{REFIab} in the previous calculations.
- If using per-bank refreshes (PER_BANK_REFRESH = 1), use t_{REFIpb} in the previous calculations.

For DDR4 mode, t_{REFI} value is different depending on the refresh mode. The appropriate value must be programmed from the specification based on the value programmed in the refresh-mode register.

Notes:
- This bitfield value must be greater than 0x1.
- If T_RFC_NOM_X1_SEL = 1, this bitfield must be greater than T_RFC_MIN in this register.
- If T_RFC_NOM_X1_SEL = 0, this bitfield * 32 must be greater than T_RFC_MIN.
- In non-DDR4 or DDR4 fixed 1x mode, this bitfield must be less than or equal to 0xFFE.
- In DDR4 fixed 2x mode, this bitfield must be less than or equal to 0x7FF.
- In DDR4 fixed 4x mode, this bitfield must be less than or equal to 0x3FF.

Unit: DFI clock cycles or multiples of 32 DFI clock cycles, depending on T_RFC_NOM_X1_SEL.
Note: Programming mode: Dynamic - Refresh related.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **T_RFC_MIN[9:0]**: t_{RFCmin}, minimum time from refresh to refresh or activate
If the DDRCTRL operates in 1:2 mode, set this bitfield to RoundUp(RoundUp(t_{RFCmin} / t_{CK}) / 2).

In LPDDR4 mode:
- If using all-bank refreshes, t_{RFCmin} in previous equations is equal to t_{RFCab}.
- If using per-bank refreshes, t_{RFCmin} in previous equations is equal to t_{RFCpb}.

In DDR4 mode, t_{RFCmin} in previous equations is different depending on the refresh mode (fixed 1x, 2x, 4x) and device density. The appropriate value must be programmed from the specification based on refresh mode and device density used.
Unit: DFI clock cycles.

Note: Programming mode: Dynamic - Refresh related.

13.7.17 DDRCTRL refresh timing1 register (DDRCTRL_REGS_RFSHTMG1)

Address offset: 0x068

Reset value: 0x008C 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_PBR2PBR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **T_PBR2PBR[7:0]**: $t_{pbR2pbR}$ for LPDDR4
Per-bank refresh to per-bank refresh different bank time.
If DDRCTRL operates in 1:2 mode, set this bitfield to RoundUp(RoundUp($t_{pbR2pbR}$ / t_{CK}) / 2).
tpbR2pbR in previous equations is different depending on the device density.
The appropriate value must be programmed from the specification.
This bitfield is valid only in LPDDR4 per-bank refresh mode (PER_BANK_REFRESH_MODE = 1 in DDRCTRL_REGS_RFSHCTL0).

Note: Programming mode: Dynamic - Refresh related.

Bits 15:0 Reserved, must be kept at reset value.

13.7.18 DDRCTRL SDRAM initialization 0 register (DDRCTRL_REGS_INIT0)

Address offset: 0x0D0

Reset value: 0x0002 004E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SKIP_DRAM_INIT[1:0]		Res.	Res.	Res.	Res.	POST_CKE_X1024[9:0]									
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PRE_CKE_X1024[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **SKIP_DRAM_INIT[1:0]**: SDRAM initialization routine

If the lower bit of this bitfield is 1, the SDRAM initialization routine is skipped. The upper bit decides in which state the controller starts up when a reset is removed.

00: The SDRAM initialization routine is run after power-up.

01: The SDRAM initialization routine is skipped after power-up. The controller starts up in normal mode.

11: The SDRAM initialization routine is skipped after power-up. The controller starts up in SR mode

Others: Reserved

Note: Programming mode: Quasi-dynamic Group 2.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:16 **POST_CKE_X1024[9:0]**: Number of cycles to wait after driving CKE high to start the SDRAM initialization sequence

LPDDR4 typically requires this bitfield to be programmed for a delay of 2 µs.

If the DDRCTRL operates in 1:2 mode, set this bitfield to JEDEC spec value divided by 2, and round it up to the next integer value.

Unit: multiples of 1024 DFI clock cycles.

Note: Programming mode: Static.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PRE_CKE_X1024[11:0]**: Number of cycles to wait after reset before driving CKE high to start the SDRAM initialization sequence

LPDDR4: t_{INIT3} of 2 ms (min)

If the DDRCTRL operates in 1:2 mode, set this bitfield to JEDEC spec value divided by 2, and round it up to the next integer value.

Unit: multiples of 1024 DFI clock cycles.

Note: Programming mode: Static.

13.7.19 DDRCTRL SDRAM initialization 1 register (DDRCTRL_REGS_INIT1)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_RSTN_X1024[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_OCD_X32[3:0]			
												rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **DRAM_RSTN_X1024[8:0]**: Number of cycles to assert SDRAM reset signal during the init sequence

This bitfield must be set to a minimum of 1. If the DDRCTRL operates in 1:2 mode, set this bitfield to JEDEC spec value divided by 2, and round it up to the next integer value.

Unit: multiples of 1024 DFI clock cycles.

Note: Programming mode: Static.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **PRE_OCD_X32[3:0]**: Wait period before driving the OCD complete command to SDRAM

Set this bitfield to zero, except if any specific requirement.

Unit: multiples of 32 DFI clock cycles.

Note: Programming mode: Static.

13.7.20 DDRCTRL SDRAM initialization 3 register (DDRCTRL_REGS_INIT3)

Address offset: 0x0DC

Reset value: 0x0000 0510

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **MR[15:0]**:

DDR3L/DDR4: Value loaded into MR0 register

LPDDR4: Value to write to MR1 register

Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bits 15:0 **EMR[15:0]**:

DDR3L/DDR4: Value to write to MR1 register (set bit 7 to 0)

LPDDR4: Value to write to MR2 register

Note: Programming mode: Quasi-dynamic Group 4.

13.7.21 DDRCTRL SDRAM initialization 4 register (DDRCTRL_REGS_INIT4)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **EMR2[15:0]**:
DDR3L/DDR4: Value to write to MR2 register
LPDDR4: Value to write to MR3 register
Note: Programming mode: Quasi-dynamic Group 4.

Bits 15:0 **EMR3[15:0]**:
DDR3L/DDR4: Value to write to MR3 register
LPDDR4: Value to write to MR13 register
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.22 DDRCTRL SDRAM initialization 5 register (DDRCTRL_REGS_INIT5)

Address offset: 0x0E4

Reset value: 0x0010 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ZQINIT_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DEV_ZQINIT_X32[7:0]**: t_{ZQINIT}, ZQ initial calibration
DDR3L typically requires 512 SDRAM clock cycles.
DDR4 requires 1024 SDRAM clock cycles.
If the DDRCTRL operates in 1:2 mode, set this bitfield to JEDEC spec value divided by 2, and round it up to the next integer value.
Unit: multiples of 32 DFI clock cycles.
Note: Programming mode: Static.

Bits 15:0 Reserved, must be kept at reset value.

13.7.23 DDRCTRL SDRAM initialization 6 register (DDRCTRL_REGS_INIT6)

Address offset: 0x0E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **MR4[15:0]**:
DDR4: Value to be loaded into SDRAM MR4 registers
LPDDR4: Value to be loaded into SDRAM MR11 registers
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:0 **MR5[15:0]**:
DDR4: Value to be loaded into SDRAM MR5 registers
LPDDR4: Value to be loaded into SDRAM MR12 registers
Note: Programming mode: Quasi-dynamic Group 1, Group 4.

13.7.24 DDRCTRL SDRAM initialization 7 register (DDRCTRL_REGS_INIT7)

Address offset: 0x0EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR22[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **MR22[15:0]**:
LPDDR4: Value to be loaded into SDRAM MR22 registers
Note: Programming mode: Quasi-dynamic Group 4.

Bits 15:0 **MR6[15:0]**:
DDR4: Value to be loaded into SDRAM MR6 registers
LPDDR4: Value to be loaded into SDRAM MR14 registers
Note: Programming mode: Quasi-dynamic Group 4.

13.7.25 DDRCTRL rank control register (DDRCTRL_REGS_RANKCTL)

Address offset: 0x0F4

Reset value: 0x0000 066F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DIFF_RANK_WR_GAP_MSB	Res.	DIFF_RANK_RD_GAP_MSB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAX_RANK_WR[3:0]				DIFF_RANK_WR_GAP[3:0]				DIFF_RANK_RD_GAP[3:0]				MAX_RANK_RD[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **DIFF_RANK_WR_GAP_MSB**:

1-bit extension to be used when DIFF_RANK_WR_GAP in this register needs to be set to a value greater than 0xF. Only present for multirank configurations

Note: Programming mode: Quasi-dynamic Group 2.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **DIFF_RANK_RD_GAP_MSB**:

1-bit extension to be used when DIFF_RANK_RD_GAP in this register needs to be set to a value greater than 0xF. Only present for multi-rank configurations

Note: Programming mode: Quasi-dynamic Group 2.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:12 **MAX_RANK_WR[3:0]**: Maximum number of writes that can be scheduled consecutively to the same rank (only present for multirank configurations)

Writes to the same rank can be performed back-to-back. Writes to different ranks require additional gap dictated by DIFF_RANK_WR_GAP in this register. This avoids possible data bus contention, and gives PHY enough time to switch the delay when changing ranks.

The DDRCTRL arbitrates bus accesses on a cycle-by-cycle basis: after a write is scheduled, there are few clock cycles (determined by DIFF_RANK_WR_GAP) in which only writes from the same rank can be scheduled. This prevents writes from other ranks from having fair access to the data bus.

After this number is reached, a delay equal to DIFF_RANK_WR_GAP is inserted by the scheduler to give all ranks a fair opportunity to be scheduled. Higher numbers increase bandwidth utilization. Lower numbers increase fairness.

This feature can be disabled by setting this bitfield to 0. In this case, the DDRCTRL stays on the same rank as long as commands are available for it.

Note: Programming mode: Static.

Bits 11:8 **DIFF_RANK_WR_GAP[3:0]**: Number of clocks of gap in data responses when performing consecutive writes to different ranks (only present for multirank configurations)

This bitfield is used to switch the delays in the PHY to match the rank requirements. This value must consider both PHY requirement and ODT requirement.

- PHY requirement ($t_{phy_wrcsgap}$)
  If write preamble is set to $2*t_{CK}$ (DDR4 only), this bitfield must be increased by 1.
  Write preamble is always set to $2*t_{CK}$ for LPDDR4.
  If write postamble is set to $1.5*t_{CK}$ (LPDDR4 only), this bitfield must be increased by 1.
- ODT requirement: This bitfield takes care of the ODT switch off timing requirement when switching ranks during writes.
  For LPDDR4, with DQ ODT enabled, this bitfield must be a minimum of ODTLoff - ODTLon - BL/2 + 1.
  For other cases, this bitfield must be a minimum of WR_ODT_HOLD - BL / 2 (in DDRCTRL_REGS_ODTCFGR).
  If the DDRCTRL operates in 1:2 mode, set this field to the larger value divided by two and round it up to the next integer.
  After PHY has completed training, the programmed value may need to be increased.
  If a value greater than 0xF is needed, DIFF_RANK_WR_GAP_MSB must be used as extension. The maximum supported value is:
- for DFI 1:2 mode, (DIFF_RANK_WR_GAP_MSB, DIFF_RANK_WR_GAP) < 0x11.
  Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2.

Bits 7:4 **DIFF_RANK_RD_GAP[3:0]**: Number of clocks of gap in data responses when performing consecutive reads to different ranks (only present for multi-rank configurations)

This bitfield is used to switch the delays in the PHY to match the rank requirements. This value must consider both PHY requirement and ODT requirement.

- PHY requirement ($t_{phy_rdcsgap}$)
  If read preamble is set to $2*t_{CK}$ (DDR4 only), this bitfield must be increased by 1.
  If read postamble is set to $1.5*t_{CK}$ (LPDDR4 only), this bitfield must be increased by 1.
- ODT requirement - This bitfield takes care of the ODT switch off timing requirement when switching ranks during reads.
  This bitfield must be a minimum of RD_ODT_HOLD - BL / 2 (in DDRCTRL_REGS_ODTCFGR).
  If the DDRCTRL operates in 1:2 mode, set this bitfield to the larger value divided by 2, and round it up to the next integer.
  After PHY has completed training, the programmed value may need to be increased.
  If a value greater than 0xF is needed, DIFF_RANK_RD_GAP_MSB must be used as extension. The maximum supported value is:
- for DFI 1:2 mode, (DIFF_RANK_WR_GAP_MSB, DIFF_RANK_WR_GAP) < 0x11.
  Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2.

Bits 3:0 **MAX_RANK_RD[3:0]**: Maximum number of reads that can be scheduled consecutively to the same rank (only present for multirank configurations)

Reads to the same rank can be performed back-to-back. Reads to different ranks require additional gap dictated by DIFF_RANK_RD_GAP in this register. This avoids possible data bus contention, and gives PHY enough time to switch the delay when changing ranks.

The DDRCTRL arbitrates bus accesses on a cycle-by-cycle basis: after a read is scheduled, there are few clock cycles (determined by DIFF_RANK_WR_GAP) in which only reads from the same rank can be scheduled. This prevents reads from other ranks from having fair access to the data bus.

After this number is reached, a delay equal to DIFF_RANK_RD_GAP is inserted by the scheduler to give all ranks a fair opportunity to be scheduled. Higher numbers increase bandwidth utilization. Lower numbers increase fairness.

This feature can be disabled by setting this bitfield to 0. In this case, the DDRCTRL stays on the same rank as long as commands are available for it.

Note: Programming mode: Static.

13.7.26 Rank control 1 register (DDRCTRL_REGS_RANKCTL1)

Address offset: 0x0F8

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR2RD_DR[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **WR2RD_DR[5:0]**: Minimum time from write command to read command on different physical ranks.

This must include time for bus turn-around between ranks and all PHY and system requirements.

After the PHY has completed training, the value programmed may need to be increased.

The following calculations are minimum values, and do not include the PHY/system requirements: WL + BL/2 - RL + WR_POSTAMBLE + RD_PREAMBLE

If using DDR4-LRDIMM, the following has to be satisfied: $t_{RPRE}/2$ + BL/2 + 2.7 - (CL - CWL)

If the calculated value is less than BL/2 (including negative values), the value should be BL/2.

When the controller is operating in 1:2 frequency ratio mode, divide the value calculated using the above equation by 2, and round it up to next integer.

For DDR4, this register value must be smaller than or equal to WR2RD_S[5:0].

For other protocols, this register value must be smaller than or equal to WR2RD[5:0].

Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

13.7.27 DDRCTRL SDRAM timing 0 register (DDRCTRL_REGS_DRAMTMG0)

Address offset: 0x100

Reset value: 0x0F10 1B0F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WR2PRE[6:0]							Res.	Res.	T_FAW[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	T_RAS_MAX[6:0]							Res.	Res.	T_RAS_MIN[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **WR2PRE[6:0]**: Minimum time between write and precharge to same bank

Specifications: WL + BL / 2 + t_{WR} = approximately 8 cycles + 15 ns = 14 clocks @400MHz and less for lower frequencies.

where:

- WL: Write latency
- BL: Burst length, must match the value programmed in the BL bit of the mode register to the SDRAM. BST (burst terminate) is not supported.
- t_{WR}: Write recovery time, comes directly from the SDRAM specification

Add one extra cycle for LPDDR4 for this parameter.

If the DDRCTRL operates in 1:2 mode, 1T mode, divide the previous value by 2 (no round-up).

If the DDRCTRL operates in 1:2 mode, 2T mode or LPDDR4 mode, divide the previous value by 2, and round it up to the next integer value.

For DDR4, LPDDR4, use nWR (WR) instead of t_{WR} to calculate the value of this bitfield.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **T_FAW[5:0]**: t_{FAW}, valid only when 8 or more banks (or banks x bank groups) are present

In 8-bank design, at most 4 banks must be activated in a rolling window of t_{FAW} cycles.

If the DDRCTRL operates in 1:2 mode, set this bitfield to t_{FAW} / 2, and round up to next integer value. In a 4-bank design, set this bitfield to 0x1.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **T_RAS_MAX[6:0]**: t_{RASmax}, max time between activate and precharge to same bank

This is the maximum time that a page can be kept open. This bitfield minimum value is 1 (0 is invalid).

If the DDRCTRL operates in 1:2 mode, this bitfield must be set to RoundDown((RoundDown(t_{RASmax} / t_{CK} / 1024) - 1) / 2).

Unit: multiples of 1024 DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **T_RAS_MIN[5:0]**: t_{RASmin}, minimum time between activate and precharge to same bank
If the DDRCTRL operates in 1:2 mode, 1T mode, set this bitfield to RoundDown(RoundUp(t_{RASmin} / t_{CK}) / 2).
If the DDRCTRL operates in 1:2 mode, 2T mode or LPDDR4 mode, set this bitfield to RoundUp(RoundUp(t_{RASmin} / t_{CK}) / 2).
Unit: DFI clock cycles.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.28 DDRCTRL SDRAM timing 1 register (DDRCTRL_REGS_DRAMTMG1)

Address offset: 0x104

Reset value: 0x0008 0414

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XP[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RD2PRE[5:0]						Res.	T_RC[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **T_XP[4:0]**: t_{XP}, minimum time after PDN exit to any operation
For DDR3L, this bitfield must be set to t_{XPDLL} if slow PDN exit is selected in MR0[12].
If C/A parity for DDR4 is used, set this bitfield to (t_{XP+PL}).
For LPDDR4 with t_{CKELPD} specified, set this bitfield to the larger of t_{XP} and t_{CKELPD}.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{XP} / 2), and round it up to the next integer value.
Units: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **RD2PRE[5:0]**: t_{RTP}, minimum time from read to precharge of same bank
For DDR3L: t_{AL} + max (RoundUp(t_{RTP} / t_{CK}), 4)
For DDR4: Max of [t_{AL} + max (RoundUp(t_{RTP} / t_{CK}), 4)]or, [RL + BL / 2 - t_{RP}].
For LPDDR4: BL / 2 + max(RoundUp(t_{RTP} / t_{CK}),8) - 8
If the DDRCTRL operates in 1:2 mode, 1T mode, divide the previous value by 2 (no round-up).
If the DDRCTRL operates in 1:2 mode, 2T mode or LPDDR4 mode, divide the previous value by 2, and round it up to the next integer value.
For DDR4, use RTP instead of t_{RTP} to calculate the value of this parameter.
Unit: DFI clock cycles
Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **T_RC[6:0]**: t_{RC}, minimum time between activates to same bank
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{RC} / 2), and round up to next integer value. Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.29 DDRCTRL SDRAM timing 2 register (DDRCTRL_REGS_DRAMTMG2)

Address offset: 0x108

Reset value: 0x0305 060D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	WRITE_LATENCY[5:0]						Res.	Res.	READ_LATENCY[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RD2WR[5:0]						Res.	Res.	WR2RD[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **WRITE_LATENCY[5:0]**: Time from write command to write data on SDRAM interface
This bitfield must be set to WL.
If the DDRCTRL operates in 1:2 mode, divide the value calculated using the previous equation by two, and round it up to next integer.
This bitfield is not required for DDR3L, as the DFI read and write latencies defined in DFITMG0 and DFITMG1 are sufficient for this protocol.
For all protocols, in addition to programming this bitfield, it is necessary to program DFITMG0 and DFITMG1 to control the read and write latencies.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **READ_LATENCY[5:0]**: Time from read command to read data on SDRAM interface
This bitfield must be set to RL.
If the DDRCTRL operates in 1:2 mode, divide the value calculated using the previous equation by two, and round it up to next integer.
This bitfield is not required for DDR3L, as the DFI read/write latencies defined in DDRCTRL_REGS_DFITMG0/1 registers are sufficient for this protocol.
For all protocols, in addition to setting this bitfield, DDRCTRL_REGS_DFITMG0/1 must be programmed to control the read and write latencies.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **RD2WR[5:0]**: Minimum time from read command to write command

This bitfield must include time for bus turnaround (both within ranks and between ranks), and all PHY and system requirements.

After the PHY has completed training, this bitfield may need to be increased.

The following calculations give minimum values (not including PHY/system requirements):

For DDR3L: RL + BL/2 + 2 - WL

For DDR4: RL + BL/2 + 1 + WR_PREAMBLE - WL

For LPDDR4 (DQ ODT disabled):

RL + BL/2 + RoundUp($t_{DQSCKmax}$ / t_{CK}) + WR_PREAMBLE + RD_POSTAMBLE - WL

For LPDDR4 (DQ ODT enabled): RL + BL/2 + RoundUp($t_{DQSCKmax}$ / t_{CK}) + RD_POSTAMBLE - ODTLon - RoundUp($t_{ODTonmin}$ / t_{CK}) + 1,

where:

- WL: Write latency
- BL: Burst length, must match the value programmed in the BL bit of the mode register to the SDRAM.
- RL: Read latency = CAS latency
- WR_PREAMBLE: 1 ($1t_{CK}$ write preamble), 2 ($2t_{CK}$ write preamble). This is unique to DDR4 and LPDDR4.
- RD_POSTAMBLE: 0.5 ($0.5t_{CK}$ read postamble), 1.5 ($1.5t_{CK}$ read postamble). This is unique to LPDDR4.

For LPDDR4, if derating is enabled (DERATEEN.derate_enable = 1), derated $t_{DQSCKmax}$ must be used.

If the DDRCTRL operates in 1:2 mode, divide the value calculated using the previous equation by two, and round it up to next integer.

Depending on the PHY, if using LRDIMM, it may be necessary to adjust the value of this parameter to compensate for the extra cycle of latency through the LRDIMM.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WR2RD[5:0]**:

In DDR4, minimum time from write command to read command for same bank group.
In other protocols, minimum time from write command to read command. This must include time for bus turnaround and all PHY and system requirements.
After the PHY has completed training, the value programmed may need to be increased.
The following calculations give minimum values (not including PHY/system requirements):
DDR4: CWL + PL + BL / 2 + t_{WTR_L}
LPDDR4: WL + BL / 2 + t_{WTR} + 1
Others: CWL + BL / 2 + t_{WTR}
Where:

- CWL: CAS write latency
- WL: Write latency
- PL: Parity latency
- BL: Burst length, must match the value programmed in the BL bit of the mode register to the SDRAM
- t_{WTR_L}: Internal write to read command delay for same bank group. This comes directly from the SDRAM specification
- t_{WTR}: Internal write to read command delay. This comes directly from the SDRAM specification. Add one extra cycle for LPDDR4 operation.

WTR_L must be increased by one if DDR4 $2t_{CK}$ write preamble is used.
If the DDRCTRL operates in 1:2 mode, divide the value calculated using the previous equation by two, and round it up to next integer.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

13.7.30 DDRCTRL SDRAM timing 3 register (DDRCTRL_REGS_DRAMTMG3)

Address offset: 0x10C

Reset value: 0x0050 400C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	T_MRW[9:0]										Res.	Res.	T_MRD[5:4]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_MRD[3:0]				Res.	Res.	T_MOD[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:20 **T_MRW[9:0]**: Time to wait after a mode register write or read (MRW or MRR)
LPDDR4: Set this to the larger of t_{MRW} and $t_{MRWCKEL}$.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **T_MRD[5:0]**: t_{MRD}, number of cycles to wait after a mode register write or read
Depending on the connected SDRAM, t_{MRD} represents:
- DDR3L/4: Time from MRS to MRS command
- LPDDR4: Time from MRS to non-MRS command

If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{MRD} / 2), and round it up to the next integer value.
If CAL mode is enabled (DFI_T_CMD_LAT > 0 in DDRCTRL_REGS_DFITMG1), t_{CAL} (= DFI_CMD_LAT in DDRCTRL_REGS_DFITMG1) must be added to the previous calculations.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **T_MOD[9:0]**: t_{MOD}, number of cycles between load mode command and following non-load mode command (only in DDR3L and DDR4)
If CAL mode is enabled (DFI_T_CMD_LAT > 0 in DDRCTRL_REGS_DFITMG1), t_{CAL} (= DFI_CMD_LAT in DDRCTRL_REGS_DFITMG1) must be added to the previous calculations.
If the DDRCTRL operates in 1:2, set this bitfield to (t_{MOD} / 2), and round it up to the next integer value.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.31 DDRCTRL SDRAM timing 4 register (DDRCTRL_REGS_DRAMTMG4)

Address offset: 0x110

Reset value: 0x0504 0405

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	T_RCD[4:0]					Res.	Res.	Res.	Res.	T_CCD[3:0]			
			rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T_RRD[3:0]				Res.	Res.	Res.	T_RP[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **T_RCD[4:0]**: t_{RCD} - t_{AL}, minimum time from activate to read or write command to same bank
If the DDRCTRL operates in 1:2 mode, set this bitfield to ((t_{RCD} - t_{AL}) / 2), and round it up to the next integer value.
Minimum value allowed for this register is 1, which implies minimum (t_{RCD} - t_{AL}) value to be two when the DDRCTRL operates in 1:2 mode.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **T_CCD[3:0]**:
For DDR4, t_{CCD_L}, minimum time between two reads or two writes for same bank group.
For other protocols, t_{CCD}, minimum time between two reads or two writes.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{CCD_L} / 2 or t_{CCD} / 2), and round it up to the next integer value.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **T_RRD[3:0]**:
For DDR4, t_{RRD_L}, minimum time between activates from bank a to bank b for same bank.
For other protocols, t_{RRD}, minimum time between activates from bank a to bank b.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{RRD_L} / 2 or t_{RRD} / 2), and round it up to the next integer value.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **T_RP[4:0]**: t_{RP}, minimum time from single-bank precharge to activate of same bank
If the DDRCTRL operates in 1:2 mode, set this bitfield to
RoundDown(RoundUp(t_{RP} / t_{CK}) / 2) + 1.
If the DDRCTRL operates in 1:2 mode in LPDDR4, set this bitfield to
RoundUp(RoundUp(t_{RP} / t_{CK}) / 2).
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.32 DDRCTRL SDRAM timing 5 register (DDRCTRL_REGS_DRAMTMG5)

Address offset: 0x114

Reset value: 0x0505 0403

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	T_CKSRX[3:0]				Res.	T_CKSRE[6:0]						
				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	T_CKESR[5:0]						Res.	Res.	Res.	T_CKE[4:0]				
		rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **T_CKSRX[3:0]**: Time before SR exit that CK is maintained as a valid clock before issuing SRX
This bitfield specifies the clock stable time before SRX. The recommended settings are:
- LPDDR4: t_{CKCKEH}
- DDR3L and DDR4: t_{CKSRX}

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **T_CKSRE[6:0]**: Time after SR entry that CK is maintained as a valid clock
This bitfield specifies the clock disable delay after SRE. The recommended settings are:
- LPDDR4: t_{CKELCK}
- DDR3L and DDR4 (no RDIMM): Max(10 ns, 5 t_{CK})

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two and round it up to next integer.
Unit: DFI clock cycles.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **T_CKESR[5:0]**: Minimum CKE low width for SR or SRPD entry to exit timing in memory clock cycles
The recommended settings are:
- LPDDR4: Max(t_{CKE}, t_{SR})
- DDR3L and DDR4 (no RDIMM): t_{CKE} + 1

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.
Unit: DFI clock cycles.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **T_CKE[4:0]**: Minimum number of cycles of CKE high/low during PDN and SR
The recommended settings are:
- LPDDR4: Max(t_{CKE}, t_{SR})
- DDR3L and DDR4: t_{CKE}

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.33 DDRCTRL SDRAM timing 6 register (DDRCTRL_REGS_DRAMTMG6)

Address offset: 0x118

Reset value: 0x0202 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKCSX[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **T_CKCSX[3:0]**: time before clock Stop exit that CK is maintained as a valid clock before issuing clock Stop exit

This bitfield specifies the clock stable time before next command after clock Stop exit. The recommended settings for LPDDR4 is t_{XP} + 2.

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.34 DDRCTRL SDRAM timing 7 register (DDRCTRL_REGS_DRAMTMG7)

Address offset: 0x11C

Reset value: 0x0000 0202

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T_CKPDE[3:0]				Res.	Res.	Res.	Res.	T_CKPDX[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **T_CKPDE[3:0]**: Time after PDN entry that CK is maintained as a valid clock

This bitfield specifies the clock disable delay after PDE. The recommended settings are:

- LPDDR4: t_{CKELCK}.
- DDR4 SDRAM: this bitfield must be set to the same value as T_CKSRE in DDRCTRL_REGS_DRAMTMG5.

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **T_CKPDX[3:0]**: Time before PDN exit that CK is maintained as a valid clock before issuing PDX

This bitfield specifies the clock stable time before PDX. The recommended settings are:

- LPDDR4: t_{CKCKEH}.
- DDR4 SDRAM: this bitfield must be set to the same value as T_CKSRX in DDRCTRL_REGS_DRAMTMG5.

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.35 DDRCTRL SDRAM timing 8 register (DDRCTRL_REGS_DRAMTMG8)

Address offset: 0x120

Reset value: 0x0303 4405

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	T_XS_FAST_X32[6:0]							Res.	T_XS_ABORT_X32[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	T_XS_DLL_X32[6:0]							Res.	T_XS_X32[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **T_XS_FAST_X32[6:0]**: t_{XS_FAST}, exit SR to ZQCL, ZQCS and MRS (only CL, WR, RTP mode)
This is applicable to only ZQCL/ZQCS commands. Ensure this bitfield ≤ T_XS_X32.
If the DDRCTRL operates in 1:2 mode, set this bitfield to the previous value divided by two, and round it up to next integer.
Unit: multiples of 32 DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **T_XS_ABORT_X32[6:0]**: t_{XS_ABORT}, exit SR to commands not requiring a locked DLL in SR abort
Ensure this bitfield ≤ T_XS_X32.
If the DDRCTRL operates in 1:2 mode, set this bitfield to the previous value divided by two, and round it up to next integer.
Unit: multiples of 32 DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **T_XS_DLL_X32[6:0]**: t_{XSDLL}, exit SR to commands requiring a locked DLL (only for DDR3L and DDR4 SDRAMs)
If the DDRCTRL operates in 1:2 mode, set this bitfield to the previous value divided by two, and round it up to next integer.
Unit: multiples of 32 DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **T_XS_X32[6:0]**: t_{XS}, exit SR to commands not requiring a locked DLL (only for DDR3L and DDR4 SDRAMs)
If the DDRCTRL operates in 1:2 mode, set this bitfield to the previous value divided by two, and round it up to next integer.
Unit: Multiples of 32 DFI clock cycles.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.36 DDRCTRL SDRAM timing 9 register (DDRCTRL_REGS_DRAMTMG9)

Address offset: 0x124

Reset value: 0x0004 040D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DDR4_WR_PREAMBLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CCD_S[2:0]		
	rw												rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T_RRD_S[3:0]				Res.	Res.	WR2RD_S[5:0]					
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DDR4_WR_PREAMBLE**: DDR4 write preamble mode
0: $1t_{CK}$ preamble
1: $2t_{CK}$ preamble
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 29:19 Reserved, must be kept at reset value.

Bits 18:16 **T_CCD_S[2:0]**: t_{CCD_S}, minimum time between two reads/writes for different bank group
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{CCD_S} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **T_RRD_S[3:0]**: t_{RRD_S}, minimum time between activates for different bank group
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{RRD_S} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WR2RD_S[5:0]**: Minimum time from write command to read command for different bank group
This must include time for bus turn-around and all PHY and system requirements.
After the PHY has completed training, the value programmed may need to be increased.
The following calculations gives minimum values (not including PHY/system requirements):
CWL + PL + BL / 2 + t_{WTR_S}
Where:
- CWL: CAS write latency
- PL: Parity latency
- BL: Burst length, must match the value programmed in the BL bit of the mode register to the SDRAM
- t_{WTR_S}: Internal write to read command delay for different bank group. This comes directly from the SDRAM specification.

WTR_S must be increased by one if DDR4 $2t_{CK}$ write preamble is used.
If the DDRCTRL operates in 1:2 mode, divide the value calculated using the previous equation by two, and round it up to next integer.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 1, Group 2, Group 4.

13.7.37 DDRCTRL SDRAM timing 10 register (DDRCTRL_REGS_DRAMTMG10)

Address offset: 0x128

Reset value: 0x001C 180A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_SYNC_GEAR[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	T_CMD_GEAR[4:0]					Res.	Res.	Res.	Res.	T_GEAR_SETUP[1:0]		T_GEAR_HOLD[1:0]	
			rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **T_SYNC_GEAR[4:0]**: Time between MRS command and the sync pulse time
This must be even number of clocks.
For DDR4-2666 and DDR4-3200, this parameter is defined as $t_{MOD(min)}$ + 4nCK,
with $t_{MOD(min)}$ = Max(24nCK, 15 ns). The maximum value for this register is 24 + 4 = 28.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{SYNC_GEAR} / 2), and round it up to the next integer.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **T_CMD_GEAR[4:0]**: sync pulse to first valid command
For DDR4-2666 and DDR4-3200, this parameter is defined as
$t_{MOD(min)}$ = Max(24nCK, 15 ns). The maximum value for this bitfield is 24.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{CMD_GEAR} / 2), and round it up to the next integer.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **T_GEAR_SETUP[1:0]**: Geardown setup time
Minimum value of this bitfield is 1 (0 is invalid).
For DDR4-2666 and DDR4-3200, this parameter is defined as 2 clocks.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{GEAR_setup} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 1:0 **T_GEAR_HOLD[1:0]**: Geardown hold time
Minimum value of this bitfield is 1 (0 is invalid).
For DDR4-2666 and DDR4-3200, this parameter is defined as 2 clocks.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{GEAR_hold} / 2), and round it up to the next integer value.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.38 DDRCTRL SDRAM timing 11 register (DDRCTRL_REGS_DRAMTMG11)

Address offset: 0x12C

Reset value: 0x440C 021C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	POST_MPSM_GAP_X32[6:0]							Res.	Res.	Res.	T_MPX_LH[4:0]				
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	T_MPX_S[1:0]		Res.	Res.	Res.	T_CKMPE[4:0]				
						rw	rw				rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **POST_MPSM_GAP_X32[6:0]**: t_{XMPDLL}, minimum exit MPSM to commands requiring a locked DLL
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{XMPDLL} / 2), and round it up to the next integer.
Unit: multiples of 32 DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **T_MPX_LH[4:0]**: t_{MPX_LH}, minimum CS_n low hold time to CKE rising edge
If the DDRCTRL operates in 1:2 mode, set this bitfield to RoundUp(t_{MPX_LH} / 2) + 1.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **T_MPX_S[1:0]**: t_{MPX_S}, minimum time CS setup time to CKE
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{MPX_S} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **T_CKMPE[4:0]**: t_{CKMPE}, minimum valid clock requirement after MPSM entry
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{CKMPE} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.39 DDRCTRL SDRAM timing 12 register (DDRCTRL_REGS_DRAMTMG12)

Address offset: 0x130

Reset value: 0x1A02 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	T_WR_MPR[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	T_CMDCKE[1:0]	
		rw	rw	rw	rw	rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_MRD_PDA[4:0]				
											rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **T_WR_MPR[5:0]**: Cycles between MPR write and other commands (DDR4 only)
Set this bitfield to t_{MOD} + AL.
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{WR_MPR} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **T_CMDCKE[1:0]**: t_{CMDCKE}, delay from valid command to CKE input low
Set this bitfield to Max(t_{ESCKE}, t_{CMDCKE}).
If the DDRCTRL operates in 1:2 mode, divide this value by two, and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **T_MRD_PDA[4:0]**: t_{MRD_PDA}, cycle time of mode register set command in PDA mode
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{MRD_PDA} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.40 DDRCTRL SDRAM timing 13 register (DDRCTRL_REGS_DRAMTMG13)

Address offset: 0x134

Reset value: 0x1C20 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ODTLOFF[6:0]							Res.	Res.	T_CCD_MW[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_PPD[2:0]		
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **ODTLOFF[6:0]**: ODTLoff, latency from CAS-2 command to ODToff reference (only LPDDR4)
If the DDRCTRL operates in 1:2 mode, set this bitfield to (ODTLoff / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **T_CCD_MW[5:0]**: t_{CCDMW}, minimum time from write or masked write to masked write command for same bank (only LPDDR4)
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{CCDMW} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **T_PPD[2:0]**: t_{PPD}, minimum time from precharge to precharge command (only LPDDR4)
If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{PPD} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.41 DDRCTRL SDRAM timing 14 register (DDRCTRL_REGS_DRAMTMG14)

Address offset: 0x138

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T_XSR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **T_XSR[11:0]**: t_{XSR}, exit SR time to any command (only LPDDR4)
If the DDRCTRL operates in 1:2 mode, divide the previous value by two, and round it up to the next integer. The 0xFFF value is illegal for this bitfield.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.42 DDRCTRL SDRAM timing 15 register (DDRCTRL_REGS_DRAMTMG15)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_DFI_LP_T_STAB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_STAB_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EN_DFI_LP_T_STAB**:
1: Enable using t_{STAB} when exiting DFI LP. This must be set when the PHY stops the clock during DFI LP to save the maximum power.
0: Disable using t_{STAB} when exiting DFI LP.
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **T_STAB_X32[7:0]**: t_{STAB}, stabilization time
It is required in the following cases for DDR3L and DDR4 RDIMM:
- when exiting power saving mode, if the clock is stopped. After reenabling it, the clock must be stable for a time specified by t_{STAB}.
- when an input clock frequency change (DDR4)
- after issuing control words that refer to clock timing (spec: 6 µs for DDR3L, 5 µs for DDR4)

If the DDRCTRL operates in 1:2 mode, set this bitfield to recommended value divided by two, and round it up to next integer.
Unit: multiples of 32 DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.43 DDRCTRL ZQ control 0 register (DDRCTRL_REGS_ZQCTL0)

Address offset: 0x180

Reset value: 0x0200 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS_AUTO_ZQ	DIS_SRX_ZQCL	ZQ_RESISTOR_SHARED	DIS_MPSMX_ZQCL	Res.	T_ZQ_LONG_NOP[10:0]										
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	T_ZQ_SHORT_NOP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DIS_AUTO_ZQ**:

1: Disable DDRCTRL generation of ZQCS/MPC(ZQ calibration) command. ZQ_CALIB_SHORT in DDRCTRL_REGS_DBGCMD can be used instead to issue ZQ calibration request from APB module.

0: Internally generate ZQCS/MPC (ZQ calibration) commands based on T_ZQ_SHORT_INTERVAL_X1024 in DDRCTRL_REGS_ZQCTL1.

Note: Programming mode: Dynamic.

Bit 30 **DIS_SRX_ZQCL**:

1: Disable issuing a ZQCL/MPC(ZQ calibration) command at SR or SRPD exit.

0: Enable issuing a ZQCL/MPC(ZQ calibration) command at SR or SRPD exit.

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bit 29 **ZQ_RESISTOR_SHARED**:

1: Denotes that ZQ resistor is shared between ranks. ZQinit/ZQCL/ZQCS/MPC(ZQ calibration) commands are sent to one rank at a time with t_{ZQinit}, t_{ZQCL}, t_{ZQCS}, t_{ZQCAL}, and t_{ZQLAT} met between commands (so that commands to different ranks do not overlap).

0: ZQ resistor is not shared.

Note: Programming mode: Static.

Bit 28 **DIS_MPSMX_ZQCL**: Only for DDR4

1: Disable issuing a ZQCL command at MPS (maximum power saving) mode exit.

0: Enable issuing a ZQCL command at MPS mode exit.

Caution: The software can send ZQCS after exiting MPS mode.

Note: Programming mode: Static.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **T_ZQ_LONG_NOP[10:0]**: DFI clock cycles of NOP required after ZQCL (ZQ calibration long) t_{ZQoper} for DDR3L/DDR4, t_{ZQCAL} for LPDDR4. MPC(ZQ Start) command is issued to the SDRAM.

If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{ZQoper} / 2 or t_{ZQCAL} / 2), and round it up to the next integer.

Unit: DFI clock cycles

Note: Programming mode: Static.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **T_ZQ_SHORT_NOP[9:0]**: DFI clock cycles of NOP required after ZQCS (ZQ calibration short) t_{ZQCS} for DDR3L/DDR4, t_{ZQLAT} for LPDDR4. MPC(ZQ Latch) command is issued to SDRAM. If the DDRCTRL operates in 1:2 mode, set this bitfield to (t_{ZQCS} / 2 or t_{ZQLAT} / 2), and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Static.

13.7.44 DDRCTRL ZQ control 1 register (DDRCTRL_REGS_ZQCTL1)

Address offset: 0x184

Reset value: 0x0200 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	T_ZQ_RESET_NOP[9:0]										T_ZQ_SHORT_INTERVAL_X1024[19:16]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_ZQ_SHORT_INTERVAL_X1024[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:20 **T_ZQ_RESET_NOP[9:0]**: $t_{ZQReset}$, DFI clock cycles of NOP required after ZQReset (ZQ calibration reset) command is issued to SDRAM
If the DDRCTRL operates in 1:2 mode, set this bitfield to ($t_{ZQReset}$ / 2) ,and round it up to the next integer.
Unit: DFI clock cycles
Note: Programming mode: Static.

Bits 19:0 **T_ZQ_SHORT_INTERVAL_X1024[19:0]**: Average interval to wait between automatically issuing ZQCS (ZQ calibration short) and MPC(ZQ calibration) commands to DDR3L/DDR4/LPDDR4 devices
Meaningless if DIS_AUTO_ZQ = 1 in DDRCTRL_REGS_ZQCTL0.
Unit: multiples of 1024 DFI clock cycles
Note: Programming mode: Static.

13.7.45 DDRCTRL ZQ control 2 register (DDRCTRL_REGS_ZQCTL2)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ZQ_RESET**: ZQ Reset operation

Setting this bit to 1 triggers a ZQ Reset operation.

When the ZQ Reset operation is complete, the DDRCTRL automatically clears this bit.

It is recommended NOT to set this bit if in init, SR (except LPDDR4), SRPD (LPDDR4), or DPD operating modes.

For SR (except LPDDR4) or SRPD (LPDDR4), ZQ Reset is scheduled after SR (except LPDDR4) or SPRD(LPDDR4) has been exited.

For DPN, ZQ Reset is not scheduled, although ZQ_RESET_BUSY is deasserted in DDRCTRL_REGS_ZQSTAT.

Note: Programming mode: Dynamic.

13.7.46 DDRCTRL ZQ status register (DDRCTRL_REGS_ZQSTAT)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET_BUSY
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ZQ_RESET_BUSY**: ZQ Reset operation possible only if this bit is low

This bit goes high in the clock after the DDRCTRL accepts the ZQ Reset request.

It goes low when the ZQ Reset command is issued to the SDRAM, and the associated NOP period is over. It is recommended not to perform ZQ Reset commands when this bit is high.

0: AZQ Reset operation can be initiated.

1: A ZQ Reset operation is in progress.

Note: Programming mode: Dynamic.

13.7.47 DDRCTRL DFI timing 0 register (DDRCTRL_REGS_DFITMG0)

Address offset: 0x190

Reset value: 0x0702 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DFI_T_CTRL_DELAY[4:0]					DFI_RDDATA_USE_DFI_PHY_CLK	DFI_T_RDDATA_EN[6:0]						
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFI_WRDATA_USE_DFI_PHY_CLK	Res.	DFI_TPHY_WRDATA[5:0]						Res.	Res.	DFI_TPHY_WRLAT[5:0]					
rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DFI_T_CTRL_DELAY[4:0]**: DFI clock cycles after an assertion or deassertion of DFI control signals

Control signals at the PHY-DRAM interface reflect this assertion or deassertion.

If the DFI clock and the memory clock are not phase-aligned, this timing parameter must be rounded up to the next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

Bit 23 **DFI_RDDATA_USE_DFI_PHY_CLK**: dfi_rddata_en, dfi_rddata, or dfi_rddata_valid generated

This bit selects whether DFI_T_RDDATA_EN value in this register is in terms of HDR (DFI clock) or SDR (DFI PHY clock) cycles:

0: HDR (DFI clock) cycles

1: SDR (DFI PHY clock) cycles

If a Synopsys DWC gen2 multiPHY is used, this bit must be 0.

Note: Programming mode: Static.

Bits 22:16 **DFI_T_RDDATA_EN[6:0]**: Time from the assertion of a read command on the DFI interface to the assertion of dfi_rddata_en

Refer to *Section 14: DDR physical interface control (DDRPHYC)* for correct value.

This bitfield corresponds to the DFI parameter trddata_en.

Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFI_RDDATA_USE_DFI_PHY_CLK in this register

Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bit 15 **DFI_WRDATA_USE_DFI_PHY_CLK**: dfi_wrdata_en, dfi_wrdata, or dfi_wrdata_mask generated

This bit selects whether DFI_TPHY_WRLAT and DFI_TPHY_WRDATA values in this register are in terms of HDR (DFI clock) or SDR (DFI PHY clock) cycles.

0: HDR (DFI clock) cycles

1: SDR (DFI PHY clock) cycles

If a Synopsys DWC gen2 multiPHY is used, this bit must be 0.

Note: Programming mode: Static.

Bit 14 Reserved, must be kept at reset value.

Bits 13:8 **DFI_TPHY_WRDATA[5:0]**: $t_{PHY_WRDATA,}$ number of clock cycles between dfi_wrdata_en assertion and associated write data driven on dfi_wrdata

Refer to *Section 14: DDR physical interface control (DDRPHYC)* for correct value.
The maximum supported value is 8.
Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFI_WRDATA_USE_DFI_PHY_CLK in this register

Note: Programming mode: Quasi-dynamic Group 4.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DFI_TPHY_WRLAT[5:0]**: t_{PHY_WRLAT}, write latency

Number of clocks from write command to write data enable (dfi_wrdata_en).
For LPDDR4, DFI_TPHY_WRLAT > 60 is not supported.
Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFI_WRDATA_USE_DFI_PHY_CLK in this register

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.48 DDRCTRL DFI timing 1 register (DDRCTRL_REGS_DFITMG1)

Address offset: 0x194

Reset value: 0x0000 0404

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFI_T_CMD_LAT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_WRDATA_DELAY[4:0]				
rw	rw	rw	rw								rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DFI_T_DRAM_CLK_DISABLE[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_ENABLE[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 **DFI_T_CMD_LAT[3:0]**: Number of DFI PHY clock cycles between when dfi_cs assertion and when the associated command is driven

This bitfield is used for CAL mode, and must be set to 0 or t_{CAL}, which matches the CAL mode register setting in the DRAM (valid range: 0 to 8).
If the PHY can add the latency for CAL mode, this bitfield must be set to 0.
Unit: DFI PHY clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 27:21 Reserved, must be kept at reset value.

Bits 20:16 **DFI_T_WRDATA_DELAY[4:0]**: t_{WRDATA_DELAY}, DFI clock cycles between dfi_wrdata_en assertion, and when the corresponding write data transfer is completed on the DRAM bus

For more information on correct value, see PHY specification.
For DFI 3.0 PHY, set this bitfield to t_{WRDATA_DELAY} (new parameter introduced in DFI 3.0).
For DFI 2.1 PHY, set this bitfield to t_{PHY_WRDATA} + (delay of DFI write data to the DRAM).
Value to be programmed is in terms of DFI clocks, not PHY clocks.
In FREQ_RATIO = 2, divide recommended value by two, and round it up to next integer.
If DFI_WRDATA_USE_DFI_PHY_CLK = 1 in DDRCTRL_REGS_DFITMG0, add 1 to the value.
Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DFI_T_DRAM_CLK_DISABLE[4:0]**: DFI clock cycles from dfi_dram_clk_disable assertion on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary

This bitfield maintains a low value on CLK. If the DFI clock and the memory clock are not phase-aligned, this timing parameter must be rounded up to the next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DFI_T_DRAM_CLK_ENABLE[4:0]**: DFI clock cycles from dfi_dram_clk_disable deassertion on the DFI until the first valid rising edge of the clock to the DRAM memory devices, at the PHY-DRAM boundary

If the DFI clock and the memory clock are not phase aligned, this timing parameter must be rounded up to the next integer.

Unit: DFI clock cycles

Note: Programming mode: Quasi-dynamic Group 4.

13.7.49 DDRCTRL DFI low-power configuration 0 register (DDRCTRL_REGS_DFILPCFG0)

Address offset: 0x198

Reset value: 0x0700 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DFI_TLP_RESP[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFI_LP_WAKEUP_SR[3:0]				Res.	Res.	Res.	DFI_LP_EN_SR	DFI_LP_WAKEUP_PD[3:0]				Res.	Res.	Res.	DFI_LP_EN_PD
rw	rw	rw	rw				rw	rw	rw	rw	rw				rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DFI_TLP_RESP[4:0]**: t_{LP_RESP}, DFI clock cycles setting for DFI

Same value is used for both PDN, SR, DPD, and MPS modes.

Unit: DFI clock cycles

Note: Programming mode: Static.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:12 **DFI_LP_WAKEUP_SR[3:0]**: t_{LP_WAKEUP}, DFI clock cycles to drive on dfi_lp_wakeup signal when SR is entered
0000: 16 cycles
0001: 32 cycles
0010: 64 cycles
0011: 128 cycles
0100: 256 cycles
0101: 512 cycles
0110: 1024 cycles
0111: 2048 cycles
1000: 4096 cycles
1001: 8192 cycles
1010: 16384 cycles
1011: 32768 cycles
1100: 65536 cycles
1101: 131072 cycles
1110: 262144 cycles
1111: Unlimited
Unit: DFI clock cycles
Note: Programming mode: Static.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **DFI_LP_EN_SR**: DFI low-power interface handshaking during SR entry/exit
0: Disabled
1: Enabled
Note: Programming mode: Static.

Bits 7:4 **DFI_LP_WAKEUP_PD[3:0]**: t_{LP_WAKEUP}, DFI clock cycles to drive on dfi_lp_wakeup signal when PDN is entered
0000: 16 cycles
0001: 32 cycles
0010: 64 cycles
0011: 128 cycles
0100: 256 cycles
0101: 512 cycles
0110: 1024 cycles
0111: 2048 cycles
1000: 4096 cycles
1001: 8192 cycles
1010: 16384 cycles
1011: 32768 cycles
1100: 65536 cycles
1101: 131072 cycles
1110: 262144 cycles
1111: Unlimited
Unit: DFI clock cycles
Note: Programming mode: Static.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DFI_LP_EN_PD**: DFI low-power interface handshaking during PDN entry/exit
0: Disabled
1: Enabled
Note: Programming mode: Static.

13.7.50 DDRCTRL DFI update 0 register (DDRCTRL_REGS_DFIUPD0)

Address offset: 0x1A0

Reset value: 0x0040 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS_AUTO_CTRLUPD	DIS_AUTO_CTRLUPD_SRX	CTRLUPD_PRE_SRX	Res.	Res.	Res.	DFI_T_CTRLUP_MAX[9:0]									
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUP_MIN[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DIS_AUTO_CTRLUPD**:
1: Disable the automatic DFI_CTRLUPD_REQ generation. The DDRCTRL must issue this signal using CTRLUPD in DDRCTRL_REGS_DBGCMD.
0: DDRCTRL issues DFI_CTRLUPD_REQ periodically.
Note: Programming mode: Quasi-dynamic Group 3.

Bit 30 **DIS_AUTO_CTRLUPD_SRX**:
1: Disable the automatic DFI_CTRLUPD_REQ generation by the DDRCTRL at SR exit.
0: DDRCTRL issues a DFI_CTRLUPD_REQ before or after exiting SR, depending on CTRLUPD_PRE_SRX in this register.
Note: Programming mode: Static.

Bit 29 **CTRLUPD_PRE_SRX**: DFI_CTRLUPD_REQ requirements at SRX
0: Send DFI_CTRLUPD_REQ after SRX.
1: Send DFI_CTRLUPD_REQ before SRX.
If DIS_AUTO_CTRLUPD_SRX = 1, this bit has no impact (no DFI_CTRLUPD_REQ issued).
For STM32MP21xx, keep the default value 0x0.
Note: Programming mode: Static.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:16 **DFI_T_CTRLUP_MAX[9:0]**: Maximum number of DFI clock cycles that DFI_CTRLUPD_REQ signal can assert
The lowest value to assign to this bitfield is 0x40.
Unit: DFI clock cycles
Note: Programming mode: Static.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **DFI_T_CTRLUP_MIN[9:0]**: Minimum number of DFI clock cycles to which the DFI_CTRLUPD_REQ signal must be asserted

The DDRCTRL expects the PHY to respond within this time. If the PHY does not respond, the DDRCTRL deasserts DFI_CTRLUPD_REQ after DFI_T_CTRLUP_MIN + 2 cycles.

The lowest value to assign to this variable is 0x1.

Unit: DFI clock cycles

Note: Programming mode: Static.

13.7.51 DDRCTRL DFI update 1 register (DDRCTRL_REGS_DFIUPD1)

Address offset: 0x1A4

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]**: Minimum amount of time between DDRCTRL initiated DFI update requests (which is executed whenever the DDRCTRL is idle)

Set this bitfield higher to reduce the frequency of update requests, which can have a small impact on the latency of the first read request when the DDRCTRL is idle.

The minimum allowed value for this bitfield is 1.

Unit: multiples of 1024 DFI clock cycles

Note: Programming mode: Static.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]**: Maximum amount of time between DDRCTRL initiated DFI update requests

This timer resets with each update request. When the timer expires, DFI_CTRLUPD_REQ is sent, and the traffic is blocked until the DFI_CTRLUPD_ACKx is received.

The PHY can use this idle time to recalibrate the delay lines to the DLLs. The DFI controller update is also used to reset PHY FIFO pointers in case of data capture errors.

Updates are required to maintain calibration over PVT, but frequent updates may impact performance. The minimum allowed value for this bitfield is 1.

This bitfield must be greater than DFI_T_CTRLUPD_INTERVAL_MIN_X1024.

Unit: multiples of 1024 DFI clock cycles

Note: Programming mode: Static.

13.7.52 DDRCTRL DFI update 2 register (DDRCTRL_REGS_DFIUPD2)

Address offset: 0x1A8

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFI_PHYUPD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **DFI_PHYUPD_EN**: Enables the support for acknowledging PHY-initiated updates
0: Disabled
1: Enabled
Note: Programming mode: Static

Bits 30:0 Reserved, must be kept at reset value.

13.7.53 DDRCTRL DFI miscellaneous control register (DDRCTRL_REGS_DFIMISC)

Address offset: 0x1B0

Reset value: 0x0000 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DFI_FREQUENCY[4:0]					LP_OPTIMIZED_WRITE	DIS_DYN_ADR_TRI	DFI_INIT_START	CTL_IDLE_EN	Res.	DFI_DATA_CS_POLARITY	PHY_DBI_MODE	DFI_INIT_COMPLETE_EN
			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DFI_FREQUENCY[4:0]**: System operating frequency
The number of supported frequencies and the mapping of signal values to clock frequencies are defined by the PHY.
Note: Programming mode: Quasi-dynamic Group 1

Bit 7 **LP_OPTIMIZED_WRITE**:
1: LPDDR4 write DQ is set to 0xF8 if masked write with enabling DBI.
0: LPDDR4 write DQ is set to 0xFF.
Note: Programming mode: Quasi-dynamic Group 3

Bit 6 **DIS_DYN_ADR_TRI**:
1: PHY specific dynamic tristating is disabled (specific feature to some Synopsys PHYs).
0: A special IDLE command is issued on the DFI while dfi_cs is inactive state, so that the PHY can detect it to turn the command/address bus Hi-Z.
It is recommended to leave this bit to its default value
Note: Programming mode: Quasi-dynamic Group 3

Bit 5 **DFI_INIT_START**: PHY init start request
When asserted, this bit triggers the PHY init start request.
Note: Programming mode: Quasi-dynamic Group 3

Bit 4 **CTL_IDLE_EN**: Enables support of ctl_idle signal (non-DFI related pin specific to certain Synopsys PHYs)
This bit must be kept at reset value.
Note: Programming mode: Static

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DFI_DATA_CS_POLARITY**: Polarity of dfi_wrdata_cs and dfi_rddata_cs signals
0: Active low
1: Active high
Note: Programming mode: Static

Bit 1 **PHY_DBI_MODE**: DBI implemented in DDRC or PHY
0: DDRC implements DBI functionality.
1: PHY implements DBI functionality.
Note: Programming mode: Static

Bit 0 **DFI_INIT_COMPLETE_EN**: PHY initialization complete enable
When asserted, the dfi_init_complete signal can be used to trigger SDRAM initialization.
Note: Programming mode: Quasi-dynamic Group 3

13.7.54 DDRCTRL DFI timing 2 register (DDRCTRL_REGS_DFITMG2)

Address offset: 0x1B4

Reset value: 0x0000 0202

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DFI_TPHY_RDCSLAT[6:0]							Res.	Res.	DFI_TPHY_WRCSLAT[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **DFI_TPHY_RDCSLAT[6:0]**: $t_{PHY_RDCSLAT}$, DFI PHY clock cycles between read command sent on the DFI control interface, and associated dfi_rddata_cs signal asserted
Refer to *Section 14: DDR physical interface control (DDRPHYC)* for the correct value.
Unit: DFI PHY clock cycles
Note: Programming mode: Quasi-dynamic Group 2, Group 4.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DFI_TPHY_WRCSLAT[5:0]**: $t_{PHY_WRCSLAT}$, DFI PHY clock cycles between write command sent on the DFI control interface, and associated dfi_wrdata_cs signal asserted.

Refer to *Section 14: DDR physical interface control (DDRPHYC)* for the correct value.

Unit: DFI PHY clock cycles

Note: Programming mode: Quasi-dynamic Group 2, Group 4.

13.7.55 DDRCTRL DFI timing 3 register (DDRCTRL_REGS_DFITMG3)

Address offset: 0x1B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_GEARDOWN_DELAY[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DFI_T_GEARDOWN_DELAY[4:0]**: Delay from dfi_geardown_en assertion to the time of t he PHY being ready to receive commands

Refer to *Section 14: DDR physical interface control (DDRPHYC)* for the correct value.

If the DDRCTRL operates in 1:2 mode, set this bitfield to ($t_{geardown_delay}$ / 2), and round it up to the next integer.

Unit: DFI clock cycles

Note: Programming mode: Static

13.7.56 DDRCTRL DFI status register (DDRCTRL_REGS_DFISTAT)

Address offset: 0x1BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_LP_ACK	DFI_INIT_COMPLETE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DFI_LP_ACK**:

Stores the value of dfi_lp_ack input to the controller

Note: Programming mode: Dynamic

Bit 0 **DFI_INIT_COMPLETE**: Status flag that announces when DFI initialization is completed
The DFI INIT is triggered by dfi_init_start signal.This flag is polled to know when the initialization is done.
Note: Programming mode: Dynamic

13.7.57 DDRCTRL DM/DBI control register (DDRCTRL_REGS_DBICTL)

Address offset: 0x1C0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_DBI_EN	WR_DBI_EN	DM_EN
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **RD_DBI_EN**: Read DBI enable signal in DDRC
0: Read DBI disabled
1: Read DBI enabled
This bit must be set at the same value as DRAM mode register.
- DDR4: MR5 bit A12. When x4 devices are used, this signal must be set to 0.
- LPDDR4: MR3[6]

Note: Programming mode: Quasi-dynamic Group 1

Bit 1 **WR_DBI_EN**: Write DBI enable signal in DDRC
0: Write DBI disabled
1: Write DBI enabled
This bit must be set at the same value as DRAM mode register.
- DDR4: MR5 bit A11. When x4 devices are used, this signal must be set to 0.
- LPDDR4: MR3[7]

Note: Programming mode: Quasi-dynamic Group 1

Bit 0 **DM_EN**: DM enable signal in DDRC
0: DM disabled
1: DM enabled
This bit must be set at the same logical value as DRAM mode register.
- DDR4: MR5 bit A10. When x4 devices are used, this signal must be set to 0.
- LPDDR4: inverted value of MR13[5] (opposite polarity from this bit)

Note: Programming mode: Static

13.7.58 DDRCTRL DFI PHY master register (DDRCTRL_REGS_DFIPHYMSTR)

Address offset: 0x1C4

Reset value: 0x8000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFI_PHYMSTR_BLK_REF_X32[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_PHYMSTR_EN
															rw

Bits 31:24 **DFI_PHYMSTR_BLK_REF_X32[7:0]**: Maximum number of DFI clock cycles that are needed to send pending refreshes before starting SR entry process

This bitfield must be used at default value (0x80) unless Synopsys suggests to change it.
00000000: 0 DFI clock cycles (no delay)
00000001: 32 DFI clock cycles
...
11111111: 8160 DFI clock cycles
Unit: multiples of 32 DFI clock cycles

Note: Programming mode: Static

Bits 23:1 Reserved, must be kept at reset value.

Bit 0 **DFI_PHYMSTR_EN**: PHY master interface enable

0: Disabled
1: Enabled

Note: Programming mode: Static

13.7.59 DDRCTRL address map 0 register (DDRCTRL_REGS_ADDRMAP0)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_CS_BIT0[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **ADDRMAP_CS_BIT0[4:0]**:
This bitfield selects the HIF address bit used as rank address bit 0.
Valid range: 0 to 29, and 31
Internal base: 6
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 31: rank address bit 0 is set to 0.
Note: Programming mode: Static

13.7.60 DDRCTRL address map 1 register (DDRCTRL_REGS_ADDRMAP1)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_BANK_B2[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADDRMAP_BANK_B1[5:0]						Res.	Res.	ADDRMAP_BANK_B0[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **ADDRMAP_BANK_B2[5:0]**:
This bitfield selects the HIF address bit used as bank address bit 2.
Valid range: 0 to 31, and 63
Internal base: 4
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 63: bank address bit 2 is set to 0.
Note: Programming mode: Static

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDRMAP_BANK_B1[5:0]**:
This bitfield selects the HIF address bits used as bank address bit 1.
Valid range: 0 to 32, and 63
Internal base: 3
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 63: bank address bit 1 is set to 0.
Note: Programming mode: Static

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **ADDRMAP_BANK_B0[5:0]**:
This bitfield selects the HIF address bits used as bank address bit 0.
Valid range: 0 to 32, and 63
Internal base: 2
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 63: bank address bit 0 is set to 0.
Note: Programming mode: Static

13.7.61 DDRCTRL address map 2 register (DDRCTRL_REGS_ADDRMAP2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_COL_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B4[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADDRMAP_COL_B3[4:0]					Res.	Res.	Res.	Res.	ADDRMAP_COL_B2[3:0]			
			rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_COL_B5[3:0]**:
- Full bus width mode: selects the HIF address bit used as column address bit 5.
- Half bus width mode: selects the HIF address bit used as column address bit 6.
- Quarter bus width mode: selects the HIF address bit used as column address bit 7.

Valid range: 0 to 7, and 15
Internal base: 5
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 15: the column address bit is set to 0.

Note: Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_COL_B4[3:0]**:
- Full bus width mode: selects the HIF address bit used as column address bit 4.
- Half bus width mode: selects the HIF address bit used as column address bit 5.
- Quarter bus width mode: selects the HIF address bit used as column address bit 6.

Valid range: 0 to 7, and 15
Internal base: 4
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, this bitfield is set to 15: the column address bit is set to 0.

Note: Programming mode: Static

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **ADDRMAP_COL_B3[4:0]**:
- Full bus width mode: selects the HIF address bit used as column address bit 3.
- Half bus width mode: selects the HIF address bit used as column address bit 4.
- Quarter bus width mode: selects the HIF address bit used as column address bit 5.

Valid range: 0 to 7
Internal base: 3
The selected HIF address bit is determined by adding the internal base to this bitfield value.

Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_COL_B2[3:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 2.
- Half bus width mode: selects the HIF address bit used as column address bit 3.
- Quarter bus width mode: selects the HIF address bit used as column address bit 4.

Valid range: 0 to 7

Internal base: 2

The selected HIF address bit is determined by adding the internal base to this bitfield value.

Note: Programming mode: Static

13.7.62 DDRCTRL address map 3 register (DDRCTRL_REGS_ADDRMAP3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADDRMAP_COL_B9[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B8[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADDRMAP_COL_B7[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B6[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADDRMAP_COL_B9[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 9.
- Half bus width mode: selects the HIF address bit used as column address bit 11.
- Quarter bus width mode: selects the HIF address bit used as column address bit 13.

Valid range: 0 to 7 and 31

Internal base: 9

The selected HIF address bit is determined by adding the internal base to this bitfield value.

- Per JEDEC DDR3L specification, column address bit 10 is reserved for indicating auto-precharge. Hence no source address bit can be mapped to column address bit 10.
- If column bit 9 is the highest column address bit, it must map to the highest valid HIF address bit (x = the highest valid HIF address bit - internal base).
- If column bit 9 is the second highest column address bit, it must map to the second highest valid HIF address bit (x = the highest valid HIF address bit 1: internal base).
- If column bit 9 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit 2: internal base)

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **ADDRMAP_COL_B8[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 8.
- Half bus width mode: selects the HIF address bit used as column address bit 9.
- Quarter bus width mode: selects the HIF address bit used as column address bit 11.

Valid range: 0 to 7, and 31

Internal base: 8

The selected HIF address bit is determined by adding the internal base to this bitfield value.

- Per JEDEC DDR3L specification, column address bit 10 is reserved for indicating auto-precharge. Hence no source address bit can be mapped to column address bit 10.
- If column bit 8 is the second highest column address bit, it must map to the second highest valid HIF address bit (x = the highest valid HIF address bit 1: internal base).
- If column bit 8 is the third highest column address bit, it must map to the third highest valid HIF address bit (x = the highest valid HIF address bit 2: internal base).

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **ADDRMAP_COL_B7[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 7.
- Half bus width mode: selects the HIF address bit used as column address bit 8.
- Quarter bus width mode: selects the HIF address bit used as column address bit 9.

Valid range: 0 to 7, and 31

Internal base: 7

The selected HIF address bit is determined by adding the internal base to this bitfield value.

- If column bit 7 is the third highest column address bit, it must map to the third highest valid HIF address bit (x = the highest valid HIF address bit 2: internal base).

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **ADDRMAP_COL_B6[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 6.
- Half bus width mode: selects the HIF address bit used as column address bit 7.
- Quarter bus width mode: selects the HIF address bit used as column address bit 8.

Valid range: 0 to 7 and 31

Internal base: 6

The selected HIF address bit is determined by adding the internal base to this bitfield value.

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

13.7.63 DDRCTRL address map 4 register (DDRCTRL_REGS_ADDRMAP4)

Address offset: 0x210

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADDRMAP_COL_B11[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B10[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **ADDRMAP_COL_B11[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 13.
- Half bus width mode: unused (see below the value to set to make it unused)
- Quarter bus width mode: unused (see below the value to set to make it unused)

Valid range: 0 to 7 and 31

Internal base: 11

The selected HIF address bit is determined by adding the internal base to this bitfield value.

- Per JEDEC DDR3L specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.
- If column bit 11 is the highest column address bit, it must map to the highest valid HIF address bit (x = the highest valid HIF address bit: internal base).
- If column bit 11 is the second highest column address bit, it must map to the second highest valid HIF address bit (x = the highest valid HIF address bit 1: internal base).
- If column bit 11 is the third highest column address bit, it must map to the third highest valid HIF address bit (x = the highest valid HIF address bit 2: internal base).

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **ADDRMAP_COL_B10[4:0]**:

- Full bus width mode: selects the HIF address bit used as column address bit 11.
- Half bus width mode: selects the HIF address bit used as column address bit 13.
- Quarter bus width mode: unused (see below the value to set to make it unused)

Valid range: 0 to 7, and 31.

Internal base: 10

The selected HIF address bit is determined by adding the internal base to this bitfield value.

- Per JEDEC DDR3L specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.
- If column bit 10 is the highest column address bit, it must map to the highest valid HIF address bit (x = the highest valid HIF address bit: internal base).
- If column bit 10 is the second highest column address bit, it must map to the second highest valid HIF address bit (x = the highest valid HIF address bit 1: internal base).
- If column bit 10 is the third highest column address bit, it must map to the third highest valid HIF address bit (x = the highest valid HIF address bit 2: internal base).

If unused, set this bitfield to 31: the column address bit is set to 0.

Note: Programming mode: Static

13.7.64 DDRCTRL address map 5 register (DDRCTRL_REGS_ADDRMAP5)

Address offset: 0x214

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B11[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2_10[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B1[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B0[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B11[3:0]**:

This bitfield selects the HIF address bit used as row address bit 11.

Valid range: 0 to 11, and 15

Internal base: 17

The selected HIF address bit is determined by adding the internal base to this bitfield value.

If unused, set this bitfield to 15: the row address bit 11 is set to 0.

Note: Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B2_10[3:0]**:

This bitfield selects the HIF address bits used as row address bits 2 to 10.

Valid range: 0 to 11, and 15

Internal base: 8 (for row address bit 2), 9 (for row address bit 3), 10 (for row address bit 4) and so on, increasing to 16 (for row address bit 10)

The selected HIF address bit is determined by adding the internal base to this bitfield value.

When this bitfield is set to 15, values of row address bits 2 to 10 are defined by DDRCTRL_REGS_ADDRMAP9/10/11 registers.

Note: Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B1[3:0]**:
This bitfield selects the HIF address bits used as row address bit 1.
Valid range: 0 to 11
Internal base: 7
The selected HIF address bit is determined by adding the internal base to this bitfield value.
Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B0[3:0]**:
This bitfield selects the HIF address bits used as row address bit 0.
Valid range: 0 to 11
Internal base: 6
The selected HIF address bit is determined by adding the internal base to this bitfield value.
Note: Programming mode: Static

13.7.65 DDRCTRL address map 6 register (DDRCTRL_REGS_ADDRMAP6)

Address offset: 0x218

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPDDR34_3GB_6GB_12GB[2:0]			Res.	ADDRMAP_ROW_B15[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B14[3:0]			
rw	rw	rw		rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B13[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B12[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:29 **LPDDR34_3GB_6GB_12GB[2:0]**: Density size/channel for LPDDR4 SDRAM device in use
When LPDDR4 device is used:
000: No LPDDR4 SDRAM 3-, 6, or 12-Gbyte device. All addresses are valid.
001: LPDDR4 SDRAM 3-Gbyte device with x16 mode. Every address having row[14:13] = 0b11 is considered as invalid.
010: LPDDR4 SDRAM 6-Gbyte device with x16 mode, or 3-Gbyte device with byte mode. Every address having row[15:14] = 0b11 is considered as invalid.
011: LPDDR4 SDRAM 12-Gbyte device with x16 mode, or 6-Gbyte device with byte mode. Every address having row[16:15] = 0b11 is considered as invalid.
101: LPDDR4 SDRAM 12-Gbyte device with byte mode. Every address having row[17:16] = 0b11 is considered as invalid.
Note: Programming mode: Static

Bit 28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B15[3:0]**:
This bitfield selects the HIF address bit used as row address bit 15.
Valid range: 0 to 11, and 15
Internal base: 21
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 15 is set to 0.
Note: Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B14[3:0]**:
This bitfield selects the HIF address bit used as row address bit 14.
Valid range: 0 to 11, and 15
Internal base: 20
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 14 is set to 0.
Note: Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B13[3:0]**:
This bitfield selects the HIF address bit used as row address bit 13.
Valid range: 0 to 11, and 15
Internal base: 19
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 13 is set to 0.
Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B12[3:0]**:
This bitfield selects the HIF address bit used as row address bit 12
Valid range: 0 to 11, and 15
Internal base: 18
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 12 is set to 0.
Note: Programming mode: Static

13.7.66 DDRCTRL address map 7 register (DDRCTRL_REGS_ADDRMAP7)

Address offset: 0x21C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B17[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B16[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B17[3:0]**:
This bitfield selects the HIF address bit used as row address bit 17.
Valid range: 0 to 11, and 15
Internal base: 23
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 17 is set to 0 for DDR4, or set to 1 for LPDDR4 backward compatibility.
Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B16[3:0]**:
This bitfield selects the HIF address bit used as row address bit 16.
Valid range: 0 to 11, and 15
Internal base: 22
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 15: the row address bit 16 is set to 0.
Note: Programming mode: Static

13.7.67 DDRCTRL address map 8 register (DDRCTRL_REGS_ADDRMAP8)

Address offset: 0x220

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADDRMAP_BG_B1[5:0]						Res.	Res.	ADDRMAP_BG_B0[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDRMAP_BG_B1[5:0]**:
This bitfield selects the HIF address bits used as bank group address bit 1.
Valid range: 0 to 32, and 63
Internal base: 3
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 63: the bank group address bit 1 is set to 0.
Note: Programming mode: Static

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **ADDRMAP_BG_B0[5:0]**:
This bitfield selects the HIF address bits used as bank group address bit 0
Valid range: 0 to 32, and 63
Internal base: 2
The selected HIF address bit is determined by adding the internal base to this bitfield value.
If unused, set this bitfield to 63: the bank group address bit 0 is set to 0.
Note: Programming mode: Static

13.7.68 DDRCTRL address map 9 register (DDRCTRL_REGS_ADDRMAP9)

Address offset: 0x224

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B4[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B3[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B5[3:0]**:
This bitfield selects the HIF address bits used as row address bit 5.
Valid range: 0 to 11
Internal base: 11
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B4[3:0]**:
This bitfield selects the HIF address bits used as row address bit 4
Valid range: 0 to 11
Internal base: 10
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B3[3:0]**:
This bitfield selects the HIF address bits used as row address bit 3
Valid range: 0 to 11
Internal base: 9
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B2[3:0]**:
This bitfield selects the HIF address bits used as row address bit 2
Valid range: 0 to 11
Internal base: 8
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

13.7.69 DDRCTRL address map 10 register (DDRCTRL_REGS_ADDRMAP10)

Address offset: 0x228

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B9[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B8[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B7[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B6[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B9[3:0]**:
This bitfield selects the HIF address bits used as row address bit 9
Valid range: 0 to 11
Internal base: 15
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B8[3:0]**:
This bitfield selects the HIF address bits used as row address bit 8
Valid range: 0 to 11
Internal base: 14
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B7[3:0]**:
This bitfield selects the HIF address bits used as row address bit 7
Valid range: 0 to 11
Internal base: 13
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B6[3:0]**:
This bitfield selects the HIF address bits used as row address bit 6
Valid range: 0 to 11
Internal base: 12
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

13.7.70 DDRCTRL address map 11 register (DDRCTRL_REGS_ADDRMAP11)

Address offset: 0x22C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B10[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B10[3:0]**:
This bitfield selects the HIF address bits used as row address bit 10
Valid range: 0 to 11
Internal base: 16
The selected HIF address bit is determined by adding the internal base to this bitfield value.
This bitfield is used only when ADDRMAP_ROW_B2_10 = 15
in DDRCTRL_REGS_ADDRMAP5.
Note: Programming mode: Static

13.7.71 DDRCTRL ODT configuration register (DDRCTRL_REGS_ODTCFG)

Address offset: 0x240

Reset value: 0x0400 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WR_ODT_HOLD[3:0]				Res.	Res.	Res.	WR_ODT_DELAY[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RD_ODT_HOLD[3:0]				Res.	RD_ODT_DELAY[4:0]					Res.	Res.
				rw	rw	rw	rw		rw	rw	rw	rw	rw		

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **WR_ODT_HOLD[3:0]**: DFI PHY clock cycles to hold ODT for a write command
The minimum supported value is 2.
Recommended values:
DDR3L: BL8 - 0x6
DDR4: BL8 - 5 + WR_PREAMBLE with WR_PREAMBLE = 1 ($1t_{CK}$ write preamble), or 2 ($2t_{CK}$ write preamble)
Unit: DFI PHY clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **WR_ODT_DELAY[4:0]**: DFI PHY clock cycles, from issuing a write command to setting ODT values associated with this command
ODT setting must remain constant for the entire time that DQS is driven by the DDRCTRL.
Recommended values:
DDR3L: 0x0
DDR4: DFI_T_CMD_LAT in DDRCTRL_REGS_DFITMG1 (to adjust for CAL mode)
Unit: DFI PHY clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RD_ODT_HOLD[3:0]**: DFI PHY clock cycles to hold ODT for a read command
The minimum supported value is 2.
Recommended values:
DDR3L: BL8 - 0x6
DDR4: BL8 - 5 + RD_PREAMBLE with RD_PREAMBLE = 1 ($1t_{CK}$ read preamble), or 2 ($2t_{CK}$ read preamble)
Unit: DFI PHY clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **RD_ODT_DELAY[4:0]**: DFI PHY clock cycles, from issuing a read command to setting ODT values associated with that command
ODT setting must remain constant for the entire time that DQS is driven by the DDRCTRL.
Recommended values:
DDR3L: CL - CWL
DDR4: CL - CWL - RD_PREAMBLE + WR_PREAMBLE + DFI_T_CMD_LAT (to adjust for CAL mode) with
- DFI_T_CMD_LAT in DDRCTRL_REGS_DFITMG1
- WR_PREAMBLE = 1 ($1t_{CK}$ write preamble), or 2 ($2t_{CK}$ write preamble)
- RD_PREAMBLE = 1 ($1t_{CK}$ read preamble), or 2 ($2t_{CK}$ read preamble)

If (CL - CWL - RD_PREAMBLE + WR_PREAMBLE) < 0, the DDRCTRL does not support ODT for read operation.
Unit: DFI PHY clock cycles
Note: Programming mode: Quasi-dynamic Group 1, Group 4.

Bits 1:0 Reserved, must be kept at reset value.

13.7.72 DDRCTRL ODT/rank map register (DDRCTRL_REGS_ODTMAP)

Address offset: 0x244

Reset value: 0x0000 2211

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RANK1_RD_ODT[1:0]		Res.	Res.	RANK1_WR_ODT[1:0]		Res.	Res.	RANK0_RD_ODT[1:0]		Res.	Res.	RANK0_WR_ODT[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:12 **RANK1_RD_ODT[1:0]**: Remote ODTs to turn on during a read from rank 1

Each rank has a remote ODT (in the SDRAM), which can be turned on by setting the appropriate bit in this bitfield.

Rank 0 is controlled by the LSB. Rank 1 is controlled by the MSB.

For each rank, set its bit to 1 to enable its ODT.

Note: Programming mode: Static.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **RANK1_WR_ODT[1:0]**: Remote ODTs to turn on during a write to rank 1

Each rank has a remote ODT (in the SDRAM), which can be turned on by setting the appropriate bit in this bitfield.

Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, and so on.

For each rank, set its bit to 1 to enable its ODT.

Note: Programming mode: Static.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **RANK0_RD_ODT[1:0]**: Remote ODTs to turn on during a read from rank 0

Each rank has a remote ODT (in the SDRAM), which can be turned on by setting the appropriate bit in this bitfield.

Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, and so on.

For each rank, set its bit to 1 to enable its ODT.

Note: Programming mode: Static.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **RANK0_WR_ODT[1:0]**: Remote ODTs to turn on during a write to rank 0

Each rank has a remote ODT (in the SDRAM), which can be turned on by setting the appropriate bit in this bitfield.

Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, and so on.

For each rank, set its bit to 1 to enable its ODT.

Note: Programming mode: Static.

13.7.73 DDRCTRL scheduler control register (DDRCTRL_REGS_SCHED)

Address offset: 0x250

Reset value: 0x8000 101C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPT_VPRW_SCH	RDWR_IDLE_GAP[6:0]							GO2CRITICAL_HYSTERESIS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPDDR4_OPT_ACT_TIMING	Res.	Res.	LPR_NUM_ENTRIES[4:0]					AUTOPRE_RMW	Res.	Res.	OPT_WRCAM_FILL_LEVEL	RDWR_SWITCH_POLICY_SEL	PAGECLOSE	PREFER_WRITE	Res.
rw			rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	

Bit 31 **OPT_VPRW_SCH**: Optimize exVPR/exVPW scheduling.

Program to 1 can improve utilization as other traffic class can utilize bandwidth while pages for exVPR/exVPW are being prepared or ranks/banks for exVPR/exVPW are being refreshed, but it can delay execution of exVPR/exVPW due to delay of Activate by read/write command of other traffic class.

Programming Mode: Static.

0: When any exVPR/exVPW are pending on CAM, read/write command of all other traffic class are masked to be scheduled

1: When any exVPR/exVPW are pending as page-hit, read/write command of all other traffic class are masked to be scheduled.

Note: Programming mode: Static.

Bits 30:24 **RDWR_IDLE_GAP[6:0]**:

When the preferred transaction store is empty for these many clock cycles, switch to the alternate transaction store if it is non-empty.

The read transaction store (both high and low priority) is the default preferred transaction store. The write transaction store is the alternative store. When prefer write over read is set, this is reversed.

0x0: The transaction store switching happens immediately when the switching conditions become true.

Unit: DFI clock cycles

Note: Programming mode: Static.

Bits 23:16 **GO2CRITICAL_HYSTERESIS[7:0]**: Not used

Note: Programming mode: Static.

Bit 15 **LPDDR4_OPT_ACT_TIMING**: Optimized ACT timing control for LPDDR4

In LPDDR4, RD/WR/ACT takes 4 cycles. To stream read/write, there is only a 4-cycle space between reads and writes.

If ACT is scheduled out after RD/WR with 1-, 2-, or 3-cycle gap, the next RD/WR may be pushed by 1, 2, or 3 cycles, and create a gap on DQ.

When this bit is set, ACT is not scheduled out with the 1-, 2-, and 3-cycle gap.

If enabled, there may be performance impact especially for random traffic (latency/utilization).

1: Enable this feature.

0: Disable this feature.

This bit is ignored when LPDDR4 = 0 in DDRCTRL_REGS_MSTR.

Note: Programming mode: Static.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **LPR_NUM_ENTRIES[4:0]**: Number of entries in the low-priority transaction store

This bitfield value + 1 gives the number of entries in the low-priority transaction store.

(1 - this bitfield) is the number of entries available in the high-priority transaction store.

Setting this bitfield to the maximum value allocates all entries to the low-priority transaction store. Setting this bitfield to 0 allocates one entry to the low-priority transaction store, and the rest to the high-priority transaction store.

Note: Programming mode: Static.

Bit 7 **AUTOPRE_RMW**: Selection of the hif_cmd_autopre behavior if a RMW is received on HIF with hif_cmd_autopre = 1

1: Apply auto-precharge only for write part of RMW.

0: Apply auto-precharge for both read and write parts of RMW.

Note: Programming mode: Static.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OPT_WRCAM_FILL_LEVEL**: Enable the feature of optimized write CAM fill level by switching to write when write CAM reaches certain fill level set in DDRCTRL_REGS_SCHED3.WRCAM_HIGHTHRESH.

When it is enabled, write starvation timer and write transaction counter automatically are not used. They are only active when the feature is disabled.

If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.

0: Disabled

1: Enabled

Note: Programming mode: Static.

Bit 3 **RDWR_SWITCH_POLICY_SEL**: Selects read write switching policy.

1: Select "enhanced" read write switching policy

0: Select "original" read write switching policy

Note: Programming mode: Static.

Bit 2 **PAGECLOSE**:

1: The bank is kept open only while there are page hit transactions available in the CAM to this bank. The last read or write command in the CAM with a bank and page hit is executed with auto-precharge if PAGECLOSE_TIMER = 0 in DDRCTRL_REGS_SCHED1.

Even if this bit is set to 1 and PAGECLOSE_TIMER = 0, an explicit precharge (and not auto-precharge) may be issued in some cases where there is a mode switch between write and read, or between LPR and HPR.

0: The bank remains open until there is a need to close it (to open a different page, or for page timeout or refresh timeout). This is also known as open-page policy, which can be overridden by setting the per-command-autopre bit on the HIF interface (hif_cmd_autopre).

This provides a midway between open- and close-page policies.

Note: Programming mode: Quasi-dynamic Group 3.

Bit 1 **PREFER_WRITE**:

If this bit is set, the bank selector prefers writes over reads. For debug purpose only.

Note: Programming mode: Static.

Bit 0 Reserved, must be kept at reset value.

13.7.74 DDRCTRL scheduler control 1 register (DDRCTRL_REGS_SCHED1)

Address offset: 0x254

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPT_HIT_GT_HPR	PAGE_HIT_LIMIT_RD[2:0]			Res.	PAGE_HIT_LIMIT_WR[2:0]			Res.	VISIBLE_WINDOW_LIMIT_RD[2:0]			Res.	VISIBLE_WINDOW_LIMIT_WR[2:0]		
rw	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DELAY_SWITCH_WRITE[3:0]				Res.	Res.	Res.	Res.	PAGECLOSE_TIMER[7:0]							
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OPT_HIT_GT_HPR**: Optimize the priority between page-hit LPR and page-miss HPR.

This is to choose trade-off between HPR latency and total utilization.

0: Page-miss HPR has priority (default). HPR latency can be better than 1 because HPR has priority over LPR.

1: Page-hit LPR has priority. DRAM utilization can be better than 0 because number of ACT-PRE is reduced.

It is recommended to enable page-hit limiter so that once page-hit limiter is expired, HPR can have priority.

Note: Programming mode: Static.

Bits 30:28 **PAGE_HIT_LIMIT_RD[2:0]**: Page-hit limiter for read.

When certain number of read commands are scheduled out without ACT for a bank (schedule page-hit commands), all entries belonging to the bank priority are increased equal to page-hit entry even if these are page-miss so that oldest entry belonging to the bank can be served regardless of page-hit/page-miss.

The priority is reset once any ACT/PRE/AP is served to the bank.

0x0: Disable this feature

0x1: 4

0x2: 8

0x3: 16

0x4: 32

Other: Reserved

Note: Programming mode: Static.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PAGE_HIT_LIMIT_WR[2:0]**: Page-hit limiter for write.
When certain number of write commands are scheduled out without ACT for a bank (schedule page-hit commands), all entries belonging to the bank priority are increased equal to page-hit entry even if these are page-miss so that oldest entry belonging to the bank can be served regardless of page-hit/page-miss.
The priority is reset once any ACT/PRE/AP is served to the bank.
0x0: Disable this feature
0x1: 4
0x2: 8
0x3: 16
0x4: 32
Others: Reserved
Note: Programming mode: Static.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **VISIBLE_WINDOW_LIMIT_RD[2:0]**: Visible window limiter for read.
This is to prevent extreme starvation against other entries within a CAM.
Each read CAM entry has a counter, it is set to the value programmed in this register when command is pushed and is counted-down when newer read CAM entry is scheduled-out.
The counter represents starvation within RD CAM in terms of number of commands to be over taken.
When the counter reaches 0, the entry becomes expired-VPR to eliminate more starvation.
0x0: Disable this feature
0x1: 31
0x2: 63
0x3: 127
0x4: 255
Others: Reserved
Note: Programming mode: Static.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **VISIBLE_WINDOW_LIMIT_WR[2:0]**: Visible window limiter for write.
This is to prevent extreme starvation against other entries within a CAM.
Each write CAM entry has a counter, it is set to the value programmed in this register when command is pushed and is counted-down when newer write CAM entry is scheduled-out.
The counter represents starvation within WR CAM in terms of number of commands to be over taken.
When the counter reaches 0, the entry becomes expired-VPW to eliminate more starvation.
0x0: Disable this feature
0x1: 31
0x2: 63
0x3: 127
0x4: 255
Others: Reserved
Note: Programming mode: Static.

Bits 15:12 **DELAY_SWITCH_WRITE[3:0]**: Indicates the number of cycles to delay switching read to write mode when write page-hit request is there and no read page-hit request is there. Setting higher value may reduce number of read to write switching but increase read to write turn-around time.
The register indicates the number of cycles.
0: no delay
1: 2 cycles delay
2: 4 cycles delay
3: 6 cycles delay
4: 8 cycles delay
...
15: 30 cycles delay
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:0 **PAGECLOSE_TIMER[7:0]**:
This bitfield works in conjunction with PAGECLOSE in DDRCTRL_REGS_SCHED. It has meaning only if PAGECLOSE = 1.
0: An auto-precharge can be scheduled for last read or write command in the CAM with a bank and page hit. Sometimes an explicit precharge is scheduled instead of the auto-precharge (see PAGECLOSE description for more details).
> 0: An auto-precharge is not scheduled for last read or write command in the CAM with a bank and page hit. A timer is started instead, with this bitfield as initial value. There is a timer on a per bank basis. This timer decrements unless the next read or write in the CAM to a bank is a page hit. It gets reset to this bitfield value if the next read or write in the CAM to a bank is a page hit. Once the timer reached zero, an explicit precharge is attempted to be scheduled.
Unit: DFI clock cycles
Note: Programming mode: Static.

13.7.75 DDRCTRL high priority read CAM 1 register (DDRCTRL_REGS_PERFHPR1)

Address offset: 0x25C

Reset value: 0x0F00 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPR_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **HPR_XACT_RUN_LENGTH[7:0]**: Transactions serviced once HPR queue goes critical
This bitfield is the smaller of this number and the number of transactions available.
Unit: transaction
Note: Programming mode: Quasi-dynamic Group 3.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **HPR_MAX_STARVE[15:0]**: DFI clocks that HPR queue can be starved before it goes critical
The minimum valid functional value for this bitfield is 0x1. Programming it to 0x0 disables the starvation functionality.
During normal operation, this function must not be disabled as it causes excessive latencies.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 3.

13.7.76 DDRCTRL low-priority read CAM 1 register (DDRCTRL_REGS_PERFLPR1)

Address offset: 0x264

Reset value: 0x0F00 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPR_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LPR_XACT_RUN_LENGTH[7:0]**: Transactions serviced once LPR queue goes critical
This bitfield is the smaller of this number and the number of transactions available.
Unit: transaction
Note: Programming mode: Quasi-dynamic Group 3.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **LPR_MAX_STARVE[15:0]**: DFI clocks that LPR queue can be starved before it goes critical
The minimum valid functional value for this bitfield is 0x1. Programming it to 0x0 disables the starvation functionality.
During normal operation, this function must not be disabled as it causes excessive latencies.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 3.

13.7.77 DDRCTRL write CAM 1 register (DDRCTRL_REGS_PERFWR1)

Address offset: 0x26C

Reset value: 0x0F00 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **W_XACT_RUN_LENGTH[7:0]**: Transactions serviced once the WR queue goes critical
This bitfield is the smaller of this number and the number of transactions available.
Unit: transaction
Note: Programming mode: Quasi-dynamic Group 3.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **W_MAX_STARVE[15:0]**: DFI clocks that the WR queue can be starved before it goes critical
The minimum valid functional value for this bitfield is 0x1. Programming it to 0x0 disables the starvation functionality.
During normal operation, this function must not be disabled as it causes excessive latencies.
Unit: DFI clock cycles
Note: Programming mode: Quasi-dynamic Group 3.

13.7.78 Scheduler control 3 register (DDRCTRL_REGS_SCHED3)

Address offset: 0x270

Reset value: 0x0404 0208

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	RD_PGHIT_NUM_THRESH[4:0]					Res.	Res.	Res.	WR_PGHIT_NUM_THRESH[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	WRCAM_HIGHTHRESH[4:0]					Res.	Res.	Res.	WRCAM_LOWTHRESH[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **RD_PGHIT_NUM_THRESH[4:0]**: Switch to read mode once number of read page-hit request exceeds the threshold set in the register during waiting t_{W2R}.
Setting to 0 disables the feature.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **WR_PGHIT_NUM_THRESH[4:0]**: Switch to write mode once number of write page-hit request exceeds threshold set in this register during waiting delay_switch_write timeout.
Setting to 0 disables the feature.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **WRCAM_HIGHTHRESH[4:0]**: The high threshold used in optimized write CAM fill level.
When (MEMC_NO_OF_ENTRY - (number of loaded entries) < wrcam_highthresh), switch to write mode and prepare banks for write direction if no Exp-VPR or read collision is there.
wrcam_highthresh must be set to a smaller value than wrcam_lowthresh.
This feature is enabled when opt_wrcam_fill_level is 1.
Note: Programming mode: Static.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **WRCAM_LOWTHRESH[4:0]**: The low threshold used in optimized write CAM fill level.
When (MEMC_NO_OF_ENTRY - (number of loaded entries) < wrcam_lowthresh), keep to write mode and stop to prepare banks for read direction if no Exp-VPR or read collision is there.
This feature is enabled when opt_wrcam_fill_level is 1.
Note: Programming mode: Static.

13.7.79 Scheduler control 4 register (DDRCTRL_REGS_SCHED4)

Address offset: 0x274

Reset value: 0x0840 0810

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WR_PAGE_EXP_CYCLES[7:0]								RD_PAGE_EXP_CYCLES[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WR_ACT_IDLE_GAP[7:0]								RD_ACT_IDLE_GAP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **WR_PAGE_EXP_CYCLES[7:0]**: Indicates the number of cycles to keep the bank opened for write direction in read mode when both directions has request to the bank.
The register indicates the number of cycle.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 23:16 **RD_PAGE_EXP_CYCLES[7:0]**: Indicates the number of cycles to keep the bank opened for read direction in write mode when both directions has request to the bank.
The register indicates the number of cycle.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 15:8 **WR_ACT_IDLE_GAP[7:0]**: Indicates the number of cycles when write direction has no request to start preparing bank for read direction.
The register indicates the number of cycle.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

Bits 7:0 **RD_ACT_IDLE_GAP[7:0]**: Indicates the number of cycles when read direction has no request to start preparing bank for write direction.
The register indicates the number of cycle.
If RDWR_SWITCH_POLICY_SEL = 0, this register is ignored.
Note: Programming mode: Static.

13.7.80 DDRCTRL debug 0 register (DDRCTRL_REGS_DBG0)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_MAX_RANK_WR_OPT	DIS_MAX_RANK_RD_OPT	Res.	DIS_COLLISION_PAGE_OPT	Res.	DIS_ACT_BYPASS	DIS_RD_BYPASS	DIS_WC
								rw	rw		rw		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **DIS_MAX_RANK_WR_OPT**: Disable optimized max_rank_wr and max_logical_rank_wr
This bit is for debug purpose only. For normal operation, this bit must be set to 0.
Note: Programming mode: Static.

Bit 6 **DIS_MAX_RANK_RD_OPT**: Disable optimized max_rank_rd and max_logical_rank_rd .
This bit is for debug purpose only. For normal operation, this bit must be set to 0.
Note: Programming mode: Static.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **DIS_COLLISION_PAGE_OPT**:
0: The auto-precharge is disabled for the flushed command in a collision case.
Collision cases are write followed by read to the same address, read followed by write to the same address, or write followed by write to same address with DIS_WC = 1 in this register (where same address comparisons exclude the two address bits representing critical word).
For debug purpose only.
Note: Programming mode: Static.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DIS_ACT_BYPASS**:
Only present in designs supporting activate bypass.
1: Disable bypass path for high-priority read activates.
For debug purpose only.
Note: Programming mode: Static.

Bit 1 **DIS_RD_BYPASS**:
Only present in designs supporting read bypass.
1: Disable bypass path for high-priority read page hits.
For debug purpose only.
Note: Programming mode: Static.

Bit 0 **DIS_WC**:
1: Disable write combine.
For debug purpose only.
Note: Programming mode: Static.

13.7.81 DDRCTRL debug 1 register (DDRCTRL_REGS_DBG1)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_HIF	DIS_DQ
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DIS_HIF**:

1: DDRCTRL asserts the HIF command signal hif_cmd_stall. The DDRCTRL ignores hif_cmd_valid and all other associated request signals.

This bit is intended to be switched on-the-fly.

Note: Programming mode: Dynamic.

Bit 0 **DIS_DQ**:

1: DDRCTRL does not dequeue any transaction from the CAM. Bypass is also disabled. All transactions are queued in the CAM. No reads or writes are issued to SDRAM as long as this bit is asserted.

This bit can be used to prevent reads or writes being issued by the DDRCTRL, which makes it safe to modify some fields associated with reads/writes.

After setting this bit, it is strongly recommended to poll WR_DATA_PIPELINE_EMPTY and RD_DATA_PIPELINE_EMPTY in DDRCTRL_REGS_DBGCAM, before making changes to any register which affect reads and writes. This ensures that the relevant logic in the DDRC is idle. This bit is intended to be switched on-the-fly.

Note: Programming mode: Dynamic.

13.7.82 DDRCTRL CAM debug register (DDRCTRL_REGS_DBGCAM)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	WR_DATA_PIPELINE_EMPTY	RD_DATA_PIPELINE_EMPTY	Res.	DBG_WR_Q_EMPTY	DBG_RD_Q_EMPTY	DBG_STALL	Res.	Res.	DBG_W_Q_DEPTH[5:0]					
		r	r		r	r	r			r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_LPR_Q_DEPTH[5:0]						Res.	Res.	DBG_HPR_Q_DEPTH[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **WR_DATA_PIPELINE_EMPTY**: Empty write data pipeline on the DFI interface
This bit is intended to be polled at least twice after setting DIS_DQ in DDRCTRL_RGES_DBG1, to ensure that all remaining commands/data have completed.
Note: Programming mode: Dynamic.

Bit 28 **RD_DATA_PIPELINE_EMPTY**: Empty read data pipeline on the DFI interface
This bit is intended to be polled at least twice after setting DIS_DQ in DDRCTRL_RGES_DBG1, to ensure that all remaining commands/data have completed.
Note: Programming mode: Dynamic.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DBG_WR_Q_EMPTY**:
1: All write command queues and write data buffers inside the DDRC are empty.
Use-case: When the DDRCTRL enters SR using the low-power entry sequence, the following is expected:
- All commands in its queues are executed.
- Write command queue and data buffers are empty.

This bit must be 1 at that time. For debug purpose only.
Note: Programming mode: Dynamic.

Bit 25 **DBG_RD_Q_EMPTY**:
1: All read command queues and read data buffers inside DDRC are empty.
Use-case: When the DDRCTRL enters SR using the low-power entry sequence, the following is expected:
- All commands in its queues are executed.
- Write command queue and data buffers are empty.

This bit must be 1 at that time. For debug purpose only.
Note: Programming mode: Dynamic.

Bit 24 **DBG_STALL**: Stall (for debug purpose only)
Note: Programming mode: Dynamic.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DBG_W_Q_DEPTH[5:0]**: Write queue depth (for debug purpose only)
Note: Programming mode: Dynamic.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DBG_LPR_Q_DEPTH[5:0]**: Low-priority read queue depth (for debug purpose only)
Note: Programming mode: Dynamic.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DBG_HPR_Q_DEPTH[5:0]**: High-priority read queue depth (for debug purpose only)
Note: Programming mode: Dynamic.

13.7.83 DDRCTRL command debug register (DDRCTRL_REGS_DBGCMD)

Address offset: 0x30C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD	ZQ_CALIB_SHORT	Res.	Res.	RANK1_REFRESH	RANK0_REFRESH
										rs	rs			rs	rs

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CTRLUPD**: dfi_ctrlupd_req to the PHY

1: The DDRCTRL issues a dfi_ctrlupd_req to the PHY. When this request is stored in the DDRCTRL, this bit is automatically cleared. This operation must only be performed when DIS_AUTO_CTRLUPD = 1 in DDRCTRL_REGS_DFIUPD0.

Note: Programming mode: Dynamic.

Bit 4 **ZQ_CALIB_SHORT**: ZQCS/MPC command to the SDRAM

1: The DDRCTRL issues a ZQCS (ZQ calibration short)/MPC (ZQ calibration) command to the SDRAM. When this request is stored in the DDRCTRL, the bit is automatically cleared.
This operation can be performed only when DIS_AUTO_ZQ = 1 in DDRCTRL_REGS_ZQCTL0.
It is recommended NOT to set this bit if in Init, in SR (except LPDDR4), or SRPD (LPDDR4).
For SR (except LPDDR4) or SRPD (LPDDR4), ZQCS is scheduled after SR (except LPDDR4) or SPRD (LPDDR4) has been exited.

Note: Programming mode: Dynamic.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RANK1_REFRESH**: Refresh to rank 1

1: The DDRCTRL issues a refresh to rank 1.
Writing to this bit causes RANK1_REFRESH_BUSY to be set in DDRCTRL_REGS_DBGSTAT. When RANK1_REFRESH_BUSY is cleared, the command has been stored in the DDRCTRL.
For 3DS configuration, refresh is sent to rank index 1.
This operation can be performed only when DIS_AUTO_REFRESH = 1 in DDRCTRL_REGS_RFSHCTL3.
It is recommended NOT to set this bit if in Init mode.

Note: Programming mode: Dynamic.

Bit 0 **RANK0_REFRESH**: Refresh to rank 0

1: The DDRCTRL issues a refresh to rank 0.
Writing to this bit causes RANK0_REFRESH_BUSY to be set in DDRCTRL_REGS_DBGSTAT. When RANK0_REFRESH_BUSY is cleared, the command has been stored in the DDRCTRL.
For 3DS configuration, refresh is sent to rank index 1.
This operation can be performed only when DIS_AUTO_REFRESH = 1 in DDRCTRL_REGS_RFSHCTL3.
It is recommended NOT to set this bit if in Init mode.

Note: Programming mode: Dynamic.

13.7.84 DDRCTRL status debug register (DDRCTRL_REGS_DBGSTAT)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD_BUSY	ZQ_CALIB_SHORT_BUSY	Res.	Res.	RANK1_REFRESH_BUSY	RANK0_REFRESH_BUSY
										r	r			r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CTRLUPD_BUSY**: CTRLUPD operation

The device can initiate a CTRLUPD operation only if this bit is low. This bit goes high in the clock after the DDRCTRL accepts the CTRLUPD request. It goes low when the CTRLUPD operation is initiated in the DDRCTRL.

It is recommended NOT to perform CTRLUPD operations when this bit is high.

0: The device can initiate a CTRLUPD operation.

1: No CTRLUPD operation has been initiated yet in the DDRCTRL.

Note: Programming mode: Dynamic.

Bit 4 **ZQ_CALIB_SHORT_BUSY**: ZQCS operation

The device can initiate a ZQCS (ZQ calibration short) operation only if this bit is low. This bit goes high in the clock after the DDRCTRL accepts the ZQCS request. It goes low when the ZQCS operation is initiated in the DDRCTRL.

It is recommended NOT to perform ZQCS operations when this bit is high.

0: The device can initiate a ZQCS operation.

1: No ZQCS operation has been initiated yet in the DDRCTRL.

Note: Programming mode: Dynamic.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RANK1_REFRESH_BUSY**: rank1_refresh operation

The device can initiate a rank1_refresh operation (refresh operation to rank 1) only if this bit is low. This bit goes high in the clock after RANK1_REFRESH is set in DDRCTRL_REGS_DBGCMD. It goes low when rank1_refresh operation is stored in the DDRCTRL.

It is recommended NOT to perform rank1_refresh operations when this bit is high.

0: The device can initiate a rank1_refresh operation.

1: No rank1_refresh operation has been stored yet in the DDRCTRL.

Note: Programming mode: Dynamic.

Bit 0 **RANK0_REFRESH_BUSY**: rank0_refresh operation

The device can initiate a rank0_refresh operation (refresh operation to rank 1) only if this bit is low. This bit goes high in the clock after RANK0_REFRESH is set in DDRCTRL_REGS_DBGCMD. It goes low when rank0_refresh operation is stored in the DDRCTRL.

It is recommended NOT to perform rank1_refresh operations when this bit is high.

0: The device can initiate a rank0_refresh operation.

1: No rank0_refresh operation has been stored yet in the DDRCTRL.

Note: Programming mode: Dynamic.

13.7.85 DDRCTRL software programming control enable register (DDRCTRL_REGS_SWCTL)

Address offset: 0x320

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SW_DONE**: Enable quasi-dynamic register programming outside reset

0: Quasi-dynamic programming enabled. Set back this bit to 1 once programming is done.

Note: Programming mode: Dynamic.

13.7.86 DDRCTRL software programming control status register (DDRCTRL_REGS_SWSTAT)

Address offset: 0x324

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE_ACK
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SW_DONE_ACK**: Register programming done

This bit is the echo of SW_DONE in DDRCTRL_REGS_SWCTL.

Wait for SW_DONE = 1 to propagate to SW_DONE_ACK at the end of the programming sequence. This ensures that the correct register values are propagated to the destination clock domains.

Note: Programming mode: Static.

13.7.87 DDRCTRL statics write enable register (DDRCTRL_REGS_SWCTLSTATIC)

Address offset: 0x328

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_STATIC_UNLOCK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SW_STATIC_UNLOCK**: Enables static register programming outside reset

1: Static register programming enabled. Set this bit back to 0 once programming is done.

Note: Programming mode: Dynamic.

13.7.88 DDRCTRL AXI poison configuration register (DDRCTRL_REGS_POISONCFG)

Address offset: 0x36C

Reset value: 0x0011 0011

This register is common to all AXI ports.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_CLR	Res.	Res.	Res.	RD_POISON_INTR_EN	Res.	Res.	Res.	RD_POISON_SLVERR_EN
							rc_w1				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_CLR	Res.	Res.	Res.	WR_POISON_INTR_EN	Res.	Res.	Res.	WR_POISON_SLVERR_EN
							rc_w1				rw				rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RD_POISON_INTR_CLR**: Interrupt clear for read transaction poisoning
1, 2, or 3 clock cycles for correct value to propagate to the DDRCTRL logic, and to clear the interrupts. The DDRCTRL automatically clears this bit.
Note: Programming mode: Dynamic.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **RD_POISON_INTR_EN**: Interrupts for read transaction poisoning enable
1: Enable interrupts for read transaction poisoning.
Note: Programming mode: Dynamic.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RD_POISON_SLVERR_EN**: SLVERR response for read transaction poisoning enable
1: Enable SLVERR response for read transaction poisoning.
Note: Programming mode: Dynamic.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **WR_POISON_INTR_CLR**: Interrupt clear for write transaction poisoning
1, 2, or 3 clock cycles for correct value to propagate to the DDRCTRL logic, and to clear the interrupts. The DDRCTRL automatically clears this bit.
Note: Programming mode: Dynamic.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WR_POISON_INTR_EN**: Interrupts for write transaction poisoning enable
1: Enable interrupts for write transaction poisoning.
Note: Programming mode: Dynamic.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **WR_POISON_SLVERR_EN**: SLVERR response for write transaction poisoning enable
1: Enable SLVERR response for write transaction poisoning.
Note: Programming mode: Dynamic.

13.7.89 DDRCTRL AXI poison status register (DDRCTRL_REGS_POISONSTAT)

Address offset: 0x370

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_1	RD_POISON_INTR_0
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_1	WR_POISON_INTR_0
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RD_POISON_INTR_1**: Read transaction poisoning error interrupt for port 1
This bit is an APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port read address channel.
The interrupt is cleared by RD_POISON_INTR_CLR in DDRCTRL_REGS_POISONCFG, then the value is propagated to the APB clock.
Note: Programming mode: Dynamic.

Bit 16 **RD_POISON_INTR_0**: Read transaction poisoning error interrupt for port 0
This bit is an APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port read address channel.
The interrupt is cleared by RD_POISON_INTR_CLR in DDRCTRL_REGS_POISONCFG, then the value is propagated to the APB clock.
Note: Programming mode: Dynamic.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **WR_POISON_INTR_1**: Write transaction poisoning error interrupt for port 1
This bit is an APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port read address channel.
The interrupt is cleared by WR_POISON_INTR_CLR in DDRCTRL_REGS_POISONCFG, then the value is propagated to the APB clock.
Note: Programming mode: Dynamic.

Bit 0 **WR_POISON_INTR_0**: Write transaction poisoning error interrupt for port 0
This bit is an APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port read address channel.
The interrupt is cleared by WR_POISON_INTR_CLR in DDRCTRL_REGS_POISONCFG, then the value is propagated to the APB clock.
Note: Programming mode: Dynamic.

13.7.90 DDRCTRL temperature derate status register (DDRCTRL_REGS_DERATESTAT)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_TEMP_LIMIT_INTR
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DERATE_TEMP_LIMIT_INTR**: Derate temperature interrupt
This bit indicates that the LPDDR4 SDRAM temperature operating limit is exceeded. It is set to 1 when the value read from MR4[2:0] is 0b000 or 0b111. This bit is cleared by DERATE_TEMP_LIMIT_INTR_CLR in DDRCTRL_REGS_DERATECTL.
Note: Programming mode: Static.

13.7.91 DDRCTRL port status register (DDRCTRL_MP_PSTAT)

Address offset: 0x3FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_BUSY_1	WR_PORT_BUSY_0
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_PORT_BUSY_1	RD_PORT_BUSY_0
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **WR_PORT_BUSY_1**: Indicates if there are outstanding writes for AXI port 1.
Note: Programming mode: Dynamic.

Bit 16 **WR_PORT_BUSY_0**: Indicates if there are outstanding writes for AXI port 0.
Note: Programming mode: Dynamic.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **RD_PORT_BUSY_1**: Indicates if there are outstanding reads for AXI port 1.
Note: Programming mode: Dynamic.

Bit 0 **RD_PORT_BUSY_0**: Indicates if there are outstanding reads for AXI port 0.
Note: Programming mode: Dynamic.

13.7.92 DDRCTRL port common configuration register (DDRCTRL_MP_PCCFG)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL_EXP_MODE	Res.	Res.	Res.	PAGEMATCH_LIMIT	Res.	Res.	Res.	GO2CRITICAL_EN
							rw				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BL_EXP_MODE**: Burst length expansion mode

0: (default value) XPI expands every AXI burst into multiple HIF commands, using the memory burst length as a unit.

1: The XPI uses half of the memory burst length as a unit. This applies to both reads and writes.

When DATA_BUS_WIDTH = 0 in DDRCTRL_REGS_MSTR, setting this bit has no effect. This can be used in order to avoid or minimize t_ccd_l penalty in DDR4, and t_ccd_mw penalty in LPDDR4. Setting this bit is then only recommended for DDR4 or LPDDR4.

If DM_EN = 0 in DDRCTRL_REGS_DBICTL, this functionality is not supported if the DCI (data channel interleave) is enabled.

Note: Programming mode: Static.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PAGEMATCH_LIMIT**: Page match four limit

1: Limit to four the number of consecutive same page DDRC transactions that can be granted by the port arbiter, when the page match feature is enabled.

0: No limit imposed on number of consecutive same page DDRC transactions

Note: Programming mode: Static.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GO2CRITICAL_EN**:

1: (enabled) Set co_gs_go2critical_wr and co_gs_go2critical_lpr/co_gs_go2critical_hpr signals going to DDRC based on urgent input (awurgent, arurgent) coming from AXI master.

0: (disabled) co_gs_go2critical_wr and co_gs_go2critical_lpr/co_gs_go2critical_hpr signals at DDRC are driven to 0.

Note: Programming mode: Static.

13.7.93 DDRCTRL port x configuration read register (DDRCTRL_MP_PCFGR_x)

Address offset: 0x404 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRB_LOCK_THRESHOLD[3:0]				Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **RRB_LOCK_THRESHOLD[3:0]**: Specifies the RRB lock threshold in configurations that disable read data interleaving.

Threshold is specified in terms of the HIF bursts that belong to the same AXI transaction. RRB locks onto VC only when this specified number of HIF bursts are returned by DDRC. RRB lock occurs earlier in cases where the AXI transaction itself is shorter and the total number of corresponding HIF bursts are below the programmed threshold and all of them are returned by DDRC.

When N is programmed in this field, the threshold will be set to N+1 bursts. Max thresholding is up to 16 bursts.

Note: Programming mode: Static.

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **RD_PORT_PAGEMATCH_EN**: Page match feature enable

1: Enable the page match feature. If enabled, once a requesting port is granted, the port is kept granted if the following immediate commands are to the same memory page (same rank, same bank and same row). See also PAGEMATCH_LIMIT in DDRCTRL_MP_PCCFG.

Note: Programming mode: Static.

Bit 13 **RD_PORT_URGENT_EN**: AXI urgent sideband signal (arurgent) enable

1: Enables arurgent. When enabled and arurgent is asserted by the master, the port becomes the highest priority, and co_gs_go2critical_lpr/co_gs_go2critical_hpr signal to the DDRC is asserted if enabled by GO2CRITICAL_EN in DDRCTRL_MP_PCCFG.

The arurgent signal can be asserted anytime, and as long as required, independently of address handshaking (not associated with any particular command).

Note: Programming mode: Static.

Bit 12 **RD_PORT_AGING_EN**: Aging function for the read channel enable

1: Enable the aging function for the read channel of the port.

Note: Programming mode: Static.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **RD_PORT_PRIORITY[9:0]**: Initial load value of read aging counters

These counters are loaded in parallel after reset, or after each grant to corresponding port. The aging counters down count every clock cycle where the port requests but is not granted. The higher significant 5 bits of the counter set the priority of the read channel of a given port.

- The port priority increases as the higher significant 5 bits of the counter start to decrease.
- When the aging counter becomes 0, the corresponding port channel has the highest priority level (timeout condition - priority0).
- For multi-port configurations, aging counters cannot be used to set port priorities when external dynamic priority inputs (arqos) are enabled (timeout still applicable).
- For single-port configurations, aging counters are only used when they timeout (become 0) to force read-write direction switching. In this case, external dynamic priority input, arqos (for reads only), can still be used to set the DDRC read priority (two priority levels: low-priority read, LPR, and high-priority read, HPR) on a command-by-command basis.

The two LSBs of this bitfield are tied internally to 0b00.

Note: Programming mode: Static.

13.7.94 DDRCTRL port x configuration write register (DDRCTRL_MP_PCFGW_x)

Address offset: 0x408 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **WR_PORT_PAGEMATCH_EN**: Page match feature enable

1: Enable the page match feature. If enabled, once a requesting port is granted, the port is kept granted if the following immediate commands are to the same memory page (same rank, same bank and same row). See also PAGEMATCH_LIMIT in DDRCTRL_MP_PCCFG.

Note: Programming mode: Static.

Bit 13 **WR_PORT_URGENT_EN**: AXI urgent sideband signal (awurgent) enable

1: Enable awurgent. When enabled and awurgent is asserted by the master, the port becomes the highest priority, and co_gs_go2critical_wr signal to DDRC is asserted if enabled by GO2CRITICAL_EN in DDRCTRL_MP_PCCFG.

The awurgent signal can be asserted anytime and as long as required, independently of address handshaking (not associated with any particular command).

Note: Programming mode: Static.

Bit 12 **WR_PORT_AGING_EN**: Aging function enable for the write channel

1: Enable the aging function for the write channel of the port.

Note: Programming mode: Static.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **WR_PORT_PRIORITY[9:0]**: Initial load value of write aging counters

These counters are loaded in parallel after reset, or after each grant to corresponding port.

The aging counters down count every clock cycle where the port requests but is not granted.

The higher significant 5 bits of the write aging counter set the initial priority of the write channel of a given port.

- The port priority increases as the higher significant 5 bits of the counter start to decrease. When the aging counter becomes 0, the corresponding port channel has the highest priority level.
- For multi-port configurations, aging counters cannot be used to set port priorities when external dynamic priority inputs (awqos) are enabled (timeout still applicable).
- For single-port configurations, aging counters are only used when they timeout (become 0) to force read-write direction switching.

The two LSBs of this bitfield are tied internally to 0b00.

Note: Programming mode: Static.

13.7.95 DDRCTRL port x control register (DDRCTRL_MP_PCTRL_x)

Address offset: 0x490 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PORT_EN**: Enable the AXI port x.
Note: Programming mode: Dynamic.

13.7.96 DDRCTRL port x read QoS configuration 0 register (DDRCTRL_MP_PCFGQOS0_x)

Address offset: 0x494 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION1[1:0]		Res.	Res.	RQOS_MAP_REGION0[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1[3:0]			
												rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **RQOS_MAP_REGION1[1:0]**: Traffic class of region 1
00: LPR
01: VPR
10 HPR
Note: Programming mode: Quasi-dynamic Group 3.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **RQOS_MAP_REGION0[1:0]**: Traffic class of region 0
V00: LPR
01: VPR
10 HPR
Note: Programming mode: Quasi-dynamic Group 3.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **RQOS_MAP_LEVEL1[3:0]**: Separation level1 that gives the end of region 0 mapping
The start of region 0 is 0. Possible values for level1 are 0 to 13 (for dual RAQ), or 0 to 14 (for single RAQ) which corresponds to arqos.
For PA, arqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.
Note: Programming mode: Quasi-dynamic Group 3.

13.7.97 DDRCTRL port x read QoS configuration 1 register (DDRCTRL_MP_PCFGQOS1_x)

Address offset: 0x498 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RQOS_MAP_TIMEOUTR[10:0]**: Timeout value for transactions mapped to red address queue
Note: Programming mode: Quasi-dynamic Group 3.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RQOS_MAP_TIMEOUTB[10:0]**: Timeout value for transactions mapped to blue address queue
Note: Programming mode: Quasi-dynamic Group 3.

13.7.98 DDRCTRL port x write QoS configuration 0 register (DDRCTRL_MP_PCFGWQOS0_x)

Address offset: 0x49C + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0E00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2[1:0]		Res.	Res.	WQOS_MAP_REGION1[1:0]		Res.	Res.	WQOS_MAP_REGION0[1:0]	
						rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **WQOS_MAP_REGION2[1:0]**: Traffic class of region 2
00: NPW
01: VPW
When VPW support is disabled (DDRCTRL_REGS_VPW_EN = 0), and this bitfield is set to 01 (VPW), the VPW traffic is aliased to NPW traffic.
Note: Programming mode: Quasi-dynamic Group 3.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **WQOS_MAP_REGION1[1:0]**: Traffic class of region 1
00: NPW
01: VPW
When VPW support is disabled (DDRCTRL_REGS_VPW_EN = 0), and this bitfield is set to 01 (VPW), the VPW traffic is aliased to NPW traffic.
Note: Programming mode: Quasi-dynamic Group 3.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **WQOS_MAP_REGION0[1:0]**: Traffic class of region 0
V00: NPW
01: VPW
When VPW support is disabled (DDRCTRL_REGS_VPW_EN = 0), and this bitfield is set to 01 (VPW), the VPW traffic is aliased to NPW traffic.
Note: Programming mode: Quasi-dynamic Group 3.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **WQOS_MAP_LEVEL2[3:0]**: Separation level2 that indicates the end of region 1 mapping
The start of region 1 is (level1 + 1). Possible values for level2 are (level1 + 1) to 14, which corresponds to awqos. Region2 starts from (level2 + 1) up to 15.
For PA, awqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.
WQOS_MAP_LEVEL1 and WQOS_MAP_LEVEL2 must have distinct values.
Note: Programming mode: Quasi-dynamic Group 3.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WQOS_MAP_LEVEL1[3:0]**: Separation level indicating the end of region 0 mapping
The start of region 0 is 0. Possible values for level1 are 0 to 13 which corresponds to awqos.
For PA, awqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.
WQOS_MAP_LEVEL1 and WQOS_MAP_LEVEL2 must have distinct values.
Note: Programming mode: Quasi-dynamic Group 3.

13.7.99 DDRCTRL port x write QoS configuration 1 register (DDRCTRL_MP_PCFGWQOS1_x)

Address offset: 0x4A0 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **WQOS_MAP_TIMEOUT2[10:0]**: Timeout value for write transactions in region 2
Note: Programming mode: Quasi-dynamic Group 3.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **WQOS_MAP_TIMEOUT1[10:0]**: Timeout value for write transactions in region 0 and 1
Note: Programming mode: Quasi-dynamic Group 3.

13.7.100 DDRCTRL version number register (DDRCTRL_MP_DDRCTRL_VER_NUMBER)

Address offset: 0xFF0

Reset value: 0x3339 312A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VER_NUMBER[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VER_NUMBER[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **VER_NUMBER[31:0]**: Device version number
This is in ASCII format, with each byte corresponding to a character of the version number
Note: Programming mode: Static.

13.7.101 DDRCTRL version type register (DDRCTRL_MP_DDRCTRL_VER_TYPE)

Address offset: 0xFF4

Reset value: 0x6761 2A2A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VER_TYPE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VER_TYPE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **VER_TYPE[31:0]**: Device version type value

This is in ASCII format, with each byte corresponding to a character of the version type

Note: Programming mode: Static.

13.7.102 DDRCTRL register map

Table 59. DDRCTRL register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DDRCTRL_REGS_MSTR**	DEVICE_CONFIG [1:0]		Res.	Res.	Res.	Res.	ACTIVE_RANKS [1:0]		Res.	Res.	Res.	Res.	BURST_RDWR [3:0]				DLL_OFF_MODE	Res.	DATA_BUS_WIDTH [1:0]		Res.	EN_2T_TIMING_MODE	BURSTCHOP		Res.	Res.	Res.	LPDDR4	DDR4	Res.	Res.	Res.	DDR3L
	Reset value	0	0					1	1					0	1	0	0	0		0	0		0	0				0	0				1
0x004	**DDRCTRL_REGS_STAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_CAM_NOT_EMPTY	Res.	Res.	SELFREF_STATE[1:0]		Res.	Res.	SELFREF_TYPE[1:0]		Res.	OPERATING_MODE[2:0]		
	Reset value																				0			0	0			0	0		0	0	0
0x008 - 0x00C	Reserved	Reserved																															
0x010	**DDRCTRL_REGS_MRCTRL0**	MR_WR	PBA_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_ADDR [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	MR_RANK [1:0]		SW_INIT_INT	PDA_EN	MPR_EN	MR_TYPE
	Reset value	0	0															0	0	0	0							1	1	0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	**DDRCTRL_REGS_MRCTRL1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_DATA[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**DDRCTRL_REGS_MRSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDA_DONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_WR_BUSY
	Reset value																								0								0
0x01C	**DDRCTRL_REGS_MRCTRL2**	MR_DEVICE_SEL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DDRCTRL_REGS_DERATEEN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_MR4_PAUSE_FC	DERATE_MR4_TUF_DIS	Res.	RC_DERATE_VALUE[2:0]		DERATE_BYTE[3:0]					Res.	DERATE_VALUE[1:0]		DERATE_ENABLE
	Reset value																			0	0		0	0	0	0	0	0	0		0	0	0
0x024	**DDRCTRL_REGS_DERATEINT**	MR4_READ_INTERVAL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	Reserved	Reserved																															
0x02C	**DDRCTRL_REGS_DERATECTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_TEMP_LIMIT_INTR_FORCE	DERATE_TEMP_LIMIT_INTR_CLR	DERATE_TEMP_LIMIT_INTR_EN
	Reset value																														0	0	1
0x030	**DDRCTRL_REGS_PWRCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDDR4_SR_ALLOWED	DIS_CAM_DRAIN_SELFREF	STAY_IN_SELFREF	SELFREF_SW	Res.	EN_DFI_DRAM_CLK_DISABLE	Res.	POWERDOWN_EN	SELFREF_EN
	Reset value																								0	0	0	0		0		0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	**DDRCTRL_REGS_PWRTMG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TO_X32[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POWERDOWN_TO_X32[4:0]				
	Reset value									0	1	0	0	0	0	0	0												1	0	0	0	0
0x038	**DDRCTRL_REGS_HWLPCTL**	Res.	Res.	Res.	Res.	HW_LP_IDLE_X32[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LP_EXIT_IDLE_EN	HW_LP_EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0															1	1
0x03C - 0x04C	Reserved	Reserved																															
0x050	**DDRCTRL_REGS_RFSHCTL0**	REFRESH_TO_AB_X32[4:0]				Res.	Res.	Res.	Res.	REFRESH_MARGIN[3:0]				Res.	Res.	Res.	REFRESH_TO_X1_X32[4:0]					Res.	Res.	REFRESH_BURST[5:0]						Res.	PER_BANK_REFRESH	AUTO_REFAB_EN[1:0]	
	Reset value	1	0	0	0	0				0	0	1	0				1	0	0	0	0			0	0	0	0	0	0		0	0	0
0x054	**DDRCTRL_REGS_RFSHCTL1**	Res.	Res.	Res.	Res.	REFRESH_TIMER1_START_VALUE_X32[11:0]												Res.	Res.	Res.	Res.	REFRESH_TIMER0_START_VALUE_X32[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x058 - 0x05C	Reserved	Reserved																															
0x060	**DDRCTRL_REGS_RFSHCTL3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_MODE[2:0]			Res.	Res.	REFRESH_UPDATE_LEVEL	DIS_AUTO_REFRESH
	Reset value																										0	0	0			0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x064	**DDRCTRL_REGS_RFSHTMG**	T_RFC_NOM_X1_SE	Res.	Res.	Res.	T_RFC_NOM_X1_X32[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	T_RFC_MIN[9:0]									
	Reset value	0				0	0	0	0	0	1	1	0	0	0	1	0							0	0	1	0	0	0	1	1	0	0
0x068	**DDRCTRL_REGS_RFSHTMG1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_PBR2PBR[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									1	0	0	0	1	1	0	0																
0x06C - 0x0CC	Reserved	Reserved																															
0x0D0	**DDRCTRL_REGS_INIT0**	SKIP_DRAM_INIT [1:0]		Res.	Res.	Res.	Res.	POST_CKE_X1024[9:0]										Res.	Res.	Res.	Res.	PRE_CKE_X1024[11:0]											
	Reset value	0	0					0	0	0	0	0	0	0	0	1	0					0	0	0	0	0	1	0	0	1	1	1	0
0x0D4	**DDRCTRL_REGS_INIT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_RSTN_X1024[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_OCD_X32[3:0]		
	Reset value								0	0	0	0	0	0	0	0	0													0	0	0	0
0x0D8	Reserved	Reserved																															
0x0DC	**DDRCTRL_REGS_INIT3**	MR[15:0]																EMR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
0x0E0	**DDRCTRL_REGS_INIT4**	EMR2[15:0]																EMR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	**DDRCTRL_REGS_INIT5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ZQINIT_X32[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	1	0	0	0	0																
0x0E8	**DDRCTRL_REGS_INIT6**	MR4[15:0]																MR5[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0EC	**DDRCTRL_REGS_INIT7**	MR22[15:0]																MR6[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F0	Reserved	Reserved																															

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F4	**DDRCTRL_REGS_RANKCTL**	Res.	Res.	Res.	Res.	Res.	DIFF_RANK_WR_GAP_MSB	Res.	DIFF_RANK_RD_GAP_MSB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAX_RANK_WR[3:0]				DIFF_RANK_WR_GAP[3:0]				DIFF_RANK_RD_GAP[3:0]				MAX_RANK_RD[3:0]			
	Reset value						0		0									0	0	0	0	0	1	1	0	0	1	1	0	1	1	1	1
0x0F8	**DDRCTRL_REGS_RANKCTL1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR2RD_DR[5:0]					
	Reset value																											0	0	1	0	0	0
0x0FC	Reserved	Reserved																															
0x100	**DDRCTRL_REGS_DRAMTMG0**	Res.	WR2PRE[6:0]							Res.	Res.	T_FAW[5:0]						Res.	T_RAS_MAX[6:0]							Res.	Res.	T_RAS_MIN[5:0]					
	Reset value		0	0	0	1	1	1	1			0	1	0	0	0	0		0	0	1	1	0	1	1			0	0	1	1	1	1
0x104	**DDRCTRL_REGS_DRAMTMG1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XP[4:0]					Res.	Res.	RD2PRE[5:0]						Res.	T_RC[6:0]						
	Reset value												0	1	0	0	0			0	0	0	1	0	0		0	0	1	0	1	0	0
0x108	**DDRCTRL_REGS_DRAMTMG2**	Res.	Res.	WRITE_LATENCY [5:0]						Reserved	Reserved	READ_LATENCY [5:0]						Res.	Res.	RD2WR[5:0]						Res.	Res.	WR2RD[5:0]					
	Reset value			0	0	0	0	1	1			0	0	0	1	0	1			0	0	0	1	1	0			0	0	1	1	0	1
0x10C	**DDRCTRL_REGS_DRAMTMG3**	Res.	Res.	T_MRW[9:0]										Res.	Res.	T_MRD[5:0]						Res.	Res.	T_MOD[9:0]									
	Reset value			0	0	0	0	0	0	0	1	0	1			0	0	0	1	0	0			0	0	0	0	0	0	1	1	0	0
0x110	**DDRCTRL_REGS_DRAMTMG4**	Res.	Res.	Res.	T_RCD[4:0]					Res.	Res.	Res.	Res.	T_CCD[3:0]				Res.	Res.	Res.	Res.	T_RRD[3:0]				Res.	Res.	Res.	T_RP[4:0]				
	Reset value				0	0	1	0	1					0	1	0	0					0	1	0	0				0	0	1	0	1
0x114	**DDRCTRL_REGS_DRAMTMG5**	Res.	Res.	Res.	Res.	T_CKSRX [3:0]				Res.	T_CKSRE[6:0]							Res.	Res.	T_CKESR[5:0]						Res.	Res.	Res.	T_CKE[4:0]				
	Reset value					0	1	0	1		0	0	0	0	1	0	1			0	0	0	1	0	0				0	0	0	1	1
0x118	**DDRCTRL_REGS_DRAMTMG6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKCSX [3:0]			
	Reset value																													0	1	0	1
0x11C	**DDRCTRL_REGS_DRAMTMG7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKPDE [3:0]				Res.	Res.	Res.	Res.	T_CKPDX [3:0]			
	Reset value																					0	0	1	0					0	0	1	0
0x120	**DDRCTRL_REGS_DRAMTMG8**	Res.	T_XS_FAST_X32[6:0]							Res.	T_XS_ABORT_X32 [6:0]							Res.	T_XS_DLL_X32[6:0]							Res.	T_XS_X32[6:0]						
	Reset value		0	0	0	0	0	1	1		0	0	0	0	0	1	1		1	0	0	0	1	0	0		0	0	0	0	1	0	1

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x124	**DDRCTRL_REGS_DRAMTMG9**	Res.	DDR4_WR_PREAMBLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CCD_S[2:0]			Res.	Res.	Res.	Res.	T_RRD_S[3:0]				Res.	Res.	WR2RD_S[5:0]					
	Reset value		0												1	0	0					0	1	0	0			0	0	1	1	0	1
0x128	**DDRCTRL_REGS_DRAMTMG10**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_SYNC_GEAR [4:0]					Res.	Res.	Res.	T_CMD_GEAR [4:0]					Res.	Res.	Res.	Res.	T_GEAR_SETUP [1:0]		T_GEAR_HOLD [1:0]	
	Reset value												1	1	1	0	0				1	1	0	0	0					1	0	1	0
0x12C	**DDRCTRL_REGS_DRAMTMG11**	Res.	POST_MPSM_GAP_X32[6:0]							Res.	Res.	Res.	T_MPX_LH [4:0]					Res.	Res.	Res.	Res.	Res.	Res.	T_MPX_S [1:0]		Res.	Res.	Res.	T_CKMPE[4:0]				
	Reset value		1	0	0	0	1	0	0				0	1	1	0	0							1	0				1	1	1	0	0
0x130	**DDRCTRL_REGS_DRAMTMG12**	Res.	Res.	T_WR_MPR[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	T_CMDCKE [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_MRD_PDA [4:0]				
	Reset value			0	1	1	0	1	0							1	0												1	0	0	0	0
0x134	**DDRCTRL_REGS_DRAMTMG13**	Res.	ODTLOFF[6:0]							Res.	Res.	T_CCD_MW[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_PPD [2:0]		
	Reset value		0	0	1	1	1	0	0			1	0	0	0	0	0													1	0	0
0x138	**DDRCTRL_REGS_DRAMTMG14**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XSR[11:0]											
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	0
0x13C	**DDRCTRL_REGS_DRAMTMG15**	EN_DFI_LP_T_STAB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_STAB_X32[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0x140 - 0x17C	Reserved	Reserved																															
0x180	**DDRCTRL_REGS_ZQCTL0**	DIS_AUTO_ZQ	DIS_SRX_ZQCL	ZQ_RESISTOR_SHARED	DIS_MPSMX_ZQCL	Res.	T_ZQ_LONG_NOP[10:0]											Res.	Res.	Res.	Res.	Res.	Res.	T_ZQ_SHORT_NOP[9:0]										
	Reset value	0	0	0	0		0	1	0	0	0	0	0	0	0	0	0							0	0	0	1	0	0	0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	**DDRCTRL_REGS_ZQCTL1**	Res.	Res.	T_ZQ_RESET_NOP[9:0]										T_ZQ_SHORT_INTERVAL_X1024[19:0]																			
	Reset value			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x188	**DDRCTRL_REGS_ZQCTL2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET
	Reset value																																0
0x18C	**DDRCTRL_REGS_ZQSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET_BUSY
	Reset value																																0
0x190	**DDRCTRL_REGS_DFITMG0**	Res.	Res.	Res.	DFI_T_CTRL_DELAY[4:0]					DFI_RDDATA_USE_DFI_PHY_CLK	DFI_T_RDDATA_EN[6:0]							DFI_WRDATA_USE_DFI_PHY_CLK	Res.	DFI_TPHY_WRDATA[5:0]						Res.	Res.	DFI_TPHY_WRLAT[5:0]						
	Reset value				0	0	1	1	1	0	0	0	0	0	0	1	0	0		0	0	0	0	0	0			0	0	0	0	1	0
0x194	**DDRCTRL_REGS_DFITMG1**	DFI_T_CMD_LAT[3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_WRDATA_DELAY[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_DISABLE[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_ENABLE[4:0]						
	Reset value	0	0	0	0								0	0	0	0	0				0	0	1	0	0				0	0	1	0	0
0x198	**DDRCTRL_REGS_DFILPCFG0**	Res.	Res.	Res.	DFI_TLP_RESP[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_LP_WAKEUP_SR[3:0]							DFI_LP_EN_SR	DFI_LP_WAKEUP_PD[3:0]				Res.	Res.	Res.	DFI_LP_EN_PD
	Reset value				0	0	1	1	1									0	0	0	0				0	0	0	0	0				0
0x19C	Reserved	Reserved																															

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1A0	**DDRCTRL_REGS_DFIUPD0**	DIS_AUTO_CTRLUPD	DIS_AUTO_CTRLUPD_SRX	CTRLUPD_PRE_SRX	Res.	Res.	Res.	DFI_T_CTRLUP_MAX[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUP_MIN[9:0]									
	Reset value	0	0	0				0	0	0	1	0	0	0	0	0	0							0	0	0	0	0	0	0	0	1	1
0x1A4	**DDRCTRL_REGS_DFIUPD1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]									
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x1A8	**DDRCTRL_REGS_DFIUPD2**	DFI_PHYUPD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1																															
0x1AC	Reserved	Reserved																															
0x1B0	**DDRCTRL_REGS_DFIMISC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_FREQUENCY[4:0]				LP_OPTIMIZED_WRITE	DIS_DYN_ADR_TRI	DFI_INIT_START	CTL_IDLE_EN	Res.	DFI_DATA_CS_POLARITY	PHY_DBI_MODE	DFI_INIT_COMPLETE_EN
	Reset value																				0	0	0	0	0	0	1	0	0		0	0	1
0x1B4	**DDRCTRL_REGS_DFITMG2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_TPHY_RDCSLAT[6:0]						Res.	Res.	DFI_TPHY_WRCSLAT[5:0]						
	Reset value																		0	0	0	0	0	1	0			0	0	0	0	1	0
0x1B8	**DDRCTRL_REGS_DFITMG3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_GEARDOWN_DELAY[4:0]				
	Reset value																												0	0	0	0	0
0x1BC	**DDRCTRL_REGS_DFISTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_LP_ACK	DFI_INIT_COMPLETE
	Reset value																															0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C0	**DDRCTRL_REGS_DBICTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_DBI_EN	WR_DBI_EN	DM_EN
	Reset value																														0	0	1
0x1C4	**DDRCTRL_REGS_DFIPHYMSTR**	DFI_PHYMSTR_BLK_REF_X32[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_PHYMSTR_EN
	Reset value	1	0	0	0	0	0	0	0																								1
0x1C8 - 0x1FC	Reserved	Reserved																															
0x200	**DDRCTRL_REGS_ADDRMAP0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_CS_BIT0[4:0]				
	Reset value																												0	0	0	0	0
0x204	**DDRCTRL_REGS_ADDRMAP1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_BANK_B2[5:0]						Res.	Res.	ADDRMAP_BANK_B1[5:0]						Res.	Res.	ADDRMAP_BANK_B0[5:0]					
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0
0x208	**DDRCTRL_REGS_ADDRMAP2**	Res.	Res.	Res.	Res.	ADDRMAP_COL_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B4[3:0]				Res.	Res.	Res.	ADDRMAP_COL_B3[4:0]					Res.	Res.	Res.	Res.	ADDRMAP_COL_B2[3:0]			
	Reset value					0	0	0	0					0	0	0	0				0	0	0	0	0					0	0	0	0	0
0x20C	**DDRCTRL_REGS_ADDRMAP3**	Res.	Res.	Res.	ADDRMAP_COL_B9[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B8[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B7[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B6[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	0
0x210	**DDRCTRL_REGS_ADDRMAP4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_COL_B11[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B10[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x214	**DDRCTRL_REGS_ADDRMAP5**	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B11 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2_10 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B1 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B0 [3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x218	**DDRCTRL_REGS_ADDRMAP6**	LPDDR34_3GB_6GB_12GB [2:0]			Res.	ADDRMAP_ROW_B15[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B14[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B13[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B12[3:0]			
	Reset value	0	0	0		0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x21C	**DDRCTRL_REGS_ADDRMAP7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B17 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B16 [3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x220	**DDRCTRL_REGS_ADDRMAP8**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_BG_B1[5:0]						Res.	Res.	ADDRMAP_BG_B0[5:0]					
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x224	**DDRCTRL_REGS_ADDRMAP9**	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B5 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B4 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B3 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2 [3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x228	**DDRCTRL_REGS_ADDRMAP10**	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B9 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B8 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B7 [3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B6 [3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x22C	**DDRCTRL_REGS_ADDRMAP11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B10 [3:0]			
	Reset value																													0	0	0	0
0x230 - 0x23C	Reserved	Reserved																															
0x240	**DDRCTRL_REGS_ODTCFG**	Res.	Res.	Res.	Res.	WR_ODT_HOLD [3:0]				Res.	Res.	Res.	WR_ODT_DELAY [4:0]					Res.	Res.	Res.	Res.	RD_ODT_HOLD [3:0]				Res.	RD_ODT_DELAY [4:0]				Res.	Res.	
	Reset value					0	1	0	0				0	0	0	0	0					0	1	0	0		0	0	0	0	0			
0x244	**DDRCTRL_REGS_ODTMAP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RANK1_RD_ODT [1:0]		Res.	Res.	RANK1_WR_ODT [1:0]		Res.	Res.	RANK0_RD_ODT [1:0]		Res.	Res.	RANK0_WR_ODT [1:0]	
	Reset value																			1	0			1	0			0	1			0	1
0x248 - 0x24C	Reserved	Reserved																															
0x250	**DDRCTRL_REGS_SCHED**	OPT_VPRW_SCH	RDWR_IDLE_GAP [6:0]							GO2CRITICAL_ HYSTERESIS[7:0]								LPDDR4_OPT_ACT_TIMING	Res.	Res.	LPR_NUM_ENTRIES[4:0]					AUTOPRE_RMW	Res.	Res.	OPT_WRCAM_FILL_LEVEL	RDWR_SWITCH_POLICY_SEL	PAGECLOSE	PREFER_WRITE	Res.	
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			1	0	0	0	0	0			1	1	1	0	
0x254	**DDRCTRL_REGS_SCHED1**	OPT_HIT_GT_HPR	PAGE_HIT_LIMIT_RD[2:0]			Res.	PAGE_HIT_LIMIT_WR[2:0]			Res.	VISIBLE_WINDOW_LIMIT_RD[2:0]			Res.	VISIBLE_WINDOW_LIMIT_WR[2:0]			DELAY_SWITCH_WRITE[3:0]				Res.	Res.	Res.	Res.	PAGECLOSE_TIMER[7:0]							
	Reset value	0	0	0	0		0	0	0		0	0	0		0	0	0	0	0	0	1	0					0	0	0	0	0	0	0	0
0x258	Reserved	Reserved																															

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x25C	**DDRCTRL_REGS_PERFHPR1**	HPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPR_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x260	Reserved	Reserved																															
0x264	**DDRCTRL_REGS_PERFLPR1**	LPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPR_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0x268	Reserved	Reserved																															
0x26C	**DDRCTRL_REGS_PERFWR1**	W_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	W_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0x270	**DDRCTRL_REGS_SCHED3**	Res.	Res.	Res.	RD_PGHIT_NUM_THRESH[4:0]					Res.	Res.	Res.	WR_PGHIT_NUM_THRESH[4:0]					Res.	Res.	Res.	WRCAM_HIGHTHRESH[4:0]					Res.	Res.	Res.	WRCAM_LOWTHRESH[4:0]				
	Reset value				0	0	1	0	0				0	0	1	0	0				0	0	0	1	0				0	1	0	0	0
0x274	**DDRCTRL_REGS_SCHED4**	WR_PAGE_EXP_CYCLES[7:0]								RD_PAGE_EXP_CYCLES[7:0]								WR_ACT_IDLE_GAP[7:0]								RD_ACT_IDLE_GAP[7:0]							
	Reset value	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0x278 - 0x2FC	Reserved	Reserved																															
0x300	**DDRCTRL_REGS_DBG0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_MAX_RANK_WR_OPT	DIS_MAX_RANK_RD_OPT	Res.	DIS_COLLISION_PAGE_OPT	Res.	DIS_ACT_BYPASS	DIS_RD_BYPASS	DIS_WC
	Reset value																									0	0		0		0	0	0
0x304	**DDRCTRL_REGS_DBG1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_HIF	DIS_DQ
	Reset value																															0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	**DDRCTRL_REGS_DBGCAM**	Res.	Res.	WR_DATA_PIPELINE_EMPTY	RD_DATA_PIPELINE_EMPTY	Res.	DBG_WR_Q_EMPTY	DBG_RD_Q_EMPTY	DBG_STALL	Res.	Res.	DBG_W_Q_DEPTH[5:0]						Res.	Res.	DBG_LPR_Q_DEPTH[5:0]						Res.	Res.	DBG_HPR_Q_DEPTH[5:0]					
	Reset value			0	0		0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x30C	**DDRCTRL_REGS_DBGCMD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD	ZQ_CALIB_SHOR	Res.	Res.	RANK1_REFRESH	RANK0_REFRESH
	Reset value																											0	0			0	0
0x310	**DDRCTRL_REGS_DBGSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD_BUSY	ZQ_CALIB_SHORT_BUSY	Res.	Res.	RANK1_REFRESH_BUSY	RANK0_REFRESH_BUSY
	Reset value																											0	0			0	0
0x314 - 0x31C	Reserved	Reserved																															
0x320	**DDRCTRL_REGS_SWCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE
	Reset value																																1
0x324	**DDRCTRL_REGS_SWSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE_ACK
	Reset value																																1
0x328	**DDRCTRL_REGS_SWCTLSTATIC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_STATIC_UNLOCK
	Reset value																																0
0x32C - 0x368	Reserved	Reserved																															

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x36C	**DDRCTRL_REGS_POISONCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_CLR	Res.	Res.	Res.	RD_POISON_INTR_EN	Res.	Res.	Res.	RD_POISON_SLVERR_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_CLR	Res.	Res.	Res.	WR_POISON_INTR_EN	Res.	Res.	Res.	WR_POISON_SLVERR_EN
	Reset value								0				1				1								0				1				1
0x370	**DDRCTRL_REGS_POISONSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_1	RD_POISON_INTR_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_1	WR_POISON_INTR_0
	Reset value															0	0															0	0
0x374 - 0x3EC	Reserved	Reserved																															
0x3F0	**DDRCTRL_REGS_DERATESTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_TEMP_LIMIT_INTR
	Reset value																																0
0x3F4 - 0x3F8	Reserved	Reserved																															
0x3FC	**DDRCTRL_MP_PSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_BUSY_1	WR_PORT_BUSY_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_PORT_BUSY_1	RD_PORT_BUSY_0
	Reset value															0	0															0	0
0x400	**DDRCTRL_MP_PCCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL_EXP_MODE	Res.	Res.	Res.	PAGEMATCH_LIMIT	Res.	Res.	Res.	GO2CRITICAL_EN
	Reset value																								0				0				0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x404	**DDRCTRL_MP_PCFGR_0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRB_LOCK_THRESHOLD[3:0]				Res.	Res.	Res.	Res.	Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	Reset value									0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0
0x408	**DDRCTRL_MP_PCFGW_0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	Reset value																		1	0	0			0	0	0	0	0	0	0	0	0	0
0x40C - 0x48C	Reserved	Reserved																															
0x490	**DDRCTRL_MP_PCTRL_0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
	Reset value																																0
0x494	**DDRCTRL_MP_PCFGQOS0_0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION1 [1:0]		Res.	Res.	RQOS_MAP_REGION0 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1 [3:0]			
	Reset value											0	0			0	0													0	0	0	0
0x498	**DDRCTRL_MP_PCFGQOS1_0**	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]											Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x49C	**DDRCTRL_MP_PCFGWQOS0_0**	Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2 [1:0]		Res.	Res.	WQOS_MAP_REGION1 [1:0]		Res.	Res.	WQOS_MAP_REGION0 [1:0]		Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2 [3:0]		Res.	Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1 [3:0]			
	Reset value							0	0			0	0			0	0					1	1	1	0					0	0	0	0
0x4A0	**DDRCTRL_MP_PCFGWQOS1_0**	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]											Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4A4 - 0x4B0	Reserved	Reserved																															
0x4B4	**DDRCTRL_MP_PCFGR_1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRB_LOCK_THRESHOLD[3:0]				Res.	Res.	Res.	Res.	Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	Reset value									0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0
0x4B8	**DDRCTRL_MP_PCFGW_1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	Reset value																		1	0	0			0	0	0	0	0	0	0	0	0	0
0x4BC - 0x53C	Reserved	Reserved																															
0x540	**DDRCTRL_MP_PCTRL_1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
	Reset value																																0
0x544	**DDRCTRL_MP_PCFGQOS0_1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION1 [1:0]		Res.	Res.	RQOS_MAP_REGION0 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1 [3:0]			
	Reset value											0	0			0	0													0	0	0	0
0x548	**DDRCTRL_MP_PCFGQOS1_1**	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]											Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x54C	**DDRCTRL_MP_PCFGWQOS0_1**	Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2 [1:0]		Res.	Res.	WQOS_MAP_REGION1 [1:0]		Res.	Res.	WQOS_MAP_REGION0 [1:0]		Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2 [3:0]				Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1 [3:0]			
	Reset value							0	0			0	0			0	0					1	1	1	0					0	0	0	0

Table 59. DDRCTRL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x550	**DDRCTRL_MP_PCFGWQOS1_1**	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]											Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x554 - 0xFEC	Reserved	Reserved																															
0xFF0	**DDRCTRL_MP_DDRCTRL_VER_NUMBER**	VER_NUMBER[31:0]																															
	Reset value	0	0	1	1	0	0	1	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0	0	1	0	0	1	0	1	0	1	0
0xFF4	**DDRCTRL_MP_DDRCTRL_VER_TYPE**	VER_TYPE[31:0]																															
	Reset value	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0

Refer to *Section 2.4* for the register boundary addresses.

14 DDR physical interface control (DDRPHYC)

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

14.1 DDRPHYC features

- 16-bit DDR PHY compatible with JEDEC Standards for DDR4, DDR3L, LPDDR4 SDRAMs
- The DDRPHYC allows operations with SDRAMs narrower than the implemented data width.
- Operating up to 1600 MT/s with LPDDR4
- Operating up to 1600 MT/s with DDR4
- Operating up to 1600 MT/s with DDR3L
- PHY utility block (PUB) to support PHY initialization, control and calibrations
- Dual interface DFI 4.0 compliant interface to the DDRCTRL
- PHY independent, firmware-based training using an embedded calibration processor, based on specialized hardware acceleration engines, and allowing automatic periodic retraining through PHY master interface
- Three inactive idle states: DFI_LP mode (most clocks and delay lines gated), PHY inactive (leakage only), PHY retention (core power removed, most I/Os powered down, SDRAMs held in SR).
- Includes a low-jitter PLL for both PHY and SDRAM clock generation. The PLL can be bypassed to support 50 to 666 MT/s.
- Test support: BIST with at-speed loopback testing, MUX-scan ATPG (stuck-at SCAN), RX and TX asynchronous paths support a maximum bit rate of 400 Mbit/s (200 MHz), PLL lock and ZQ calibration test
- Programmable output and ODT impedance with dynamic PVT compensation

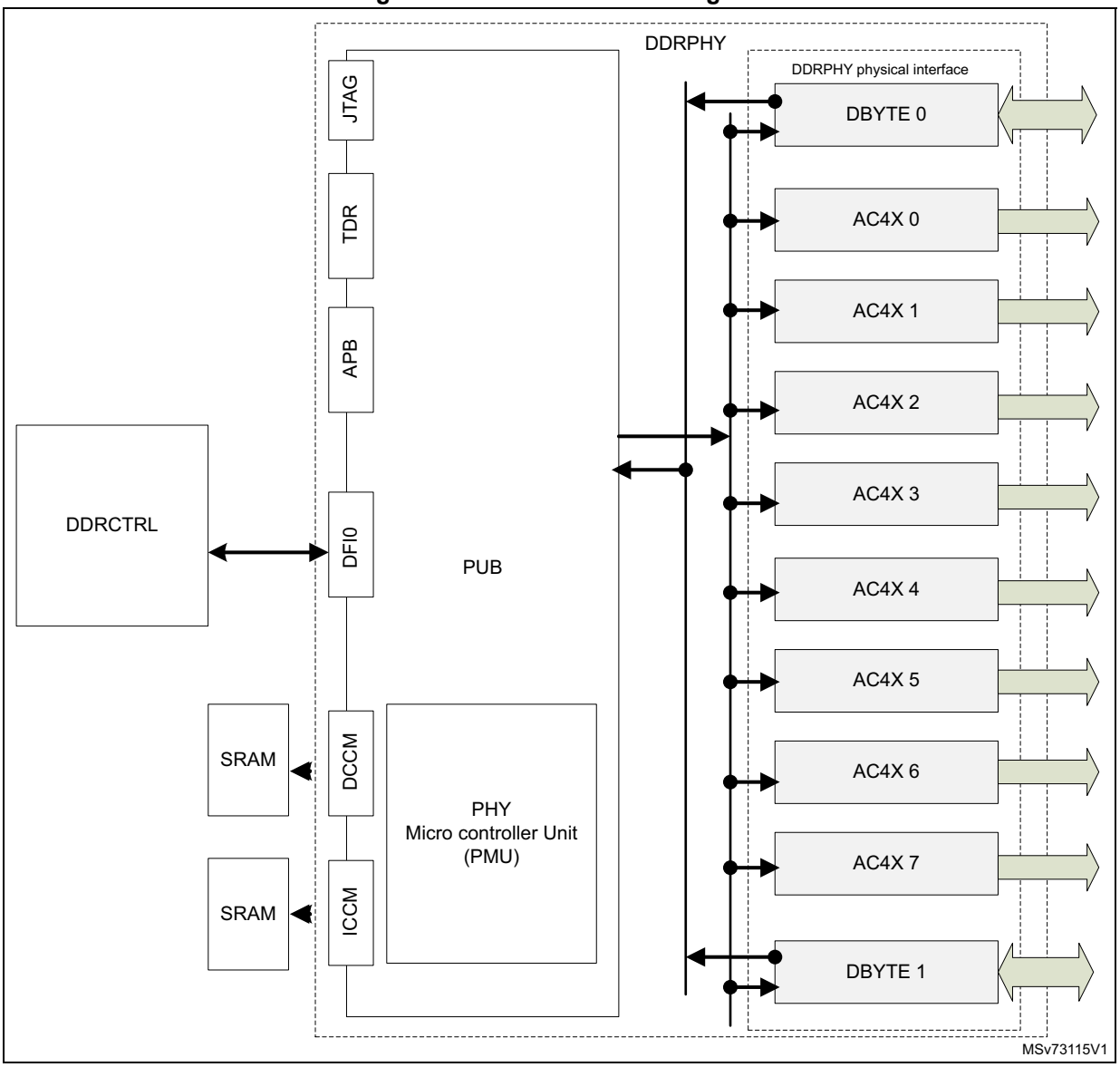
14.2 DDRPHYC block diagram

The DDRPHYC block diagram is shown in *Figure 40*.

The DDRPHYC is composed of:

- a DDRPHY physical interface
- a PUB which is used to service the PHY. It is controlled via the APB and JTAG interfaces. This PUB embeds a microcontroller as training engine, and interfaces with a local coupled memory for training firmware and data.

Figure 40. DDRPHYC block diagram



14.3 DDRPHYC SDRAM interface

The DDRPHYC SDRAM interfaces are listed in the table below.

Table 60. Function of DDRPHYC pins versus DDR protocol

Pin name	Pin function according to protocol			Comment
	LPDDR4	DDR4	DDR3L	
DDR_RESETN	RESET_N	RESET_N	RESET_N	-
-	MTEST1	ALERT/MTEST1	ERR/MTEST1	Not available on STM32MP21xx
DDR_VREF	-	-	-	Not used(1)

Table 60. Function of DDRPHYC pins versus DDR protocol (continued)

Pin name	Pin function according to protocol			Comment
	LPDDR4	DDR4	DDR3L	
DDR_ZQ	-	-	-	240 Ω 1% to GND
DDR_A0	CKEA0	CKE0	CKE0	-
DDR_A1	-	CKE1	CKE1	CKEA1 and CKE1 are not supported in STM32MP21xx.
DDR_A2	CAA[0]	CS_N0	CS_N0	-
DDR_A3	CAA[1]	ODT0	ODT0	-
DDR_A4	CLKA_T	CLK0_T	CLK0_T	-
DDR_A5	CLKA_C	CLK0_C	CLK0_C	-
DDR_A6	CSA0	CS_N1	CS_N1	CS_N1 is not supported in STM32MP21xx.
DDR_A7	-	ODT1	ODT1	CAS1 and ODT1 are not supported in STM32MP21xx.
DDR_A8	CAA[2]	A[9]	A[9]	-
DDR_A9	CAA[3]	A[12]	A[12]	-
DDR_A10	CAA[4]	A[11]	A[11]	-
DDR_A11	CAA[5]	A[7]	A[7]	-
DDR_A12	-	A[8]	A[8]	-
DDR_A13	-	A[6]	A[6]	CKEB1 is not supported in STM32MP21xx.
DDR_A14	-	A[5]	A[5]	-
DDR_A15	-	A[4]	A[4]	-
DDR_A16	-	BG[1]	BA[2]	BG1 is supported with DDR4 x8 addressing.
DDR_A17	-	ACT_N	A[14]	-
DDR_A18	-	BG[0]	A[15]	-
DDR_A19	-	PAR	PAR	CSB1 and PAR are not supported in STM32MP21xx.
DDR_A20	-	A[3]	A[3]	-
DDR_A21	-	A[2]	A[2]	-
DDR_A22	-	A[1]	A[1]	-
DDR_A23	-	BA[1]	BA[1]	-
DDR_A25	-	A[13]	A[13]	-
DDR_A26	-	BA[0]	BA[0]	-
DDR_A27	-	A[10]	A[10]	-
DDR_A28	-	A[0]	A[0]	-

Table 60. Function of DDRPHYC pins versus DDR protocol (continued)

Pin name	Pin function according to protocol			Comment
	LPDDR4	DDR4	DDR3L	
DDR_A29	-	CAS_N/A[15]	CAS_N	-
DDR_A30	-	WE_N/A[14]	WE_N	-
DDR_A31	-	RAS_N/A[16]	RAS_N	-
DDR_DQ[7] to DDR_DQ[0]	Any to any DQ mapping within the byte lane is supported.			-
DDR_DM0	DMI0_A	DMBI0_n	DML	-
DDR_DQS0P	DQS0_t_A	DQS0_t	DQSL	-
DDR_DQS0N	DQS0_c_A	DQS0_c	DQSL#	-
DDR_DQ[15] to DDR_DQ[8]	Any to any DQ mapping within the byte lane is supported.			-
DDR_DM1	DMI1_A	DMBI1_n	DMU	-
DDR_DQS1P	DQS1_t_A	DQS1_t	DQSU	-
DDR_DQS1N	DQS1_c_A	DQS1_c	DQSU#	-

1. For observation only.

14.4 DDRPHY functional overview

The DDRPHY includes the following components:

- MASTER: contains the PLL, as well as observation and debug signals
- ACX4: responsible for the address and command bus
- DBYTE: responsible for the DQ bus data transfers and timing

14.4.1 MASTER

The MASTER block contains the PLL, the primary voltage reference, thermal calibration, reset, and I/O cells for test and control.

- PLL

  The PLL takes the DFICLK (generated from the RCC PLL2) as an input, and creates an internal clock by quadrupling the DFICLK frequency. The PLL in the MASTER can only multiply the input clock frequency to produce an output clock that is four times the frequency of the input clock. The internal clock is the output of the MASTER to the ACX4 and DBYTE.PLL. Control signals are controlled through registers in the PUB.

- Calibration

  The MASTER contains the analog blocks to perform output drive and input termination impedance calibration. Using a digital control state machine, located in the PUB, the full mixed-signal calibration loop matches the impedance of the driver and termination with the impedance of an external precision resistor. The resulting digital calibration code is fed to the address/command, DQ, and DQS blocks to control their impedance.

- Reset

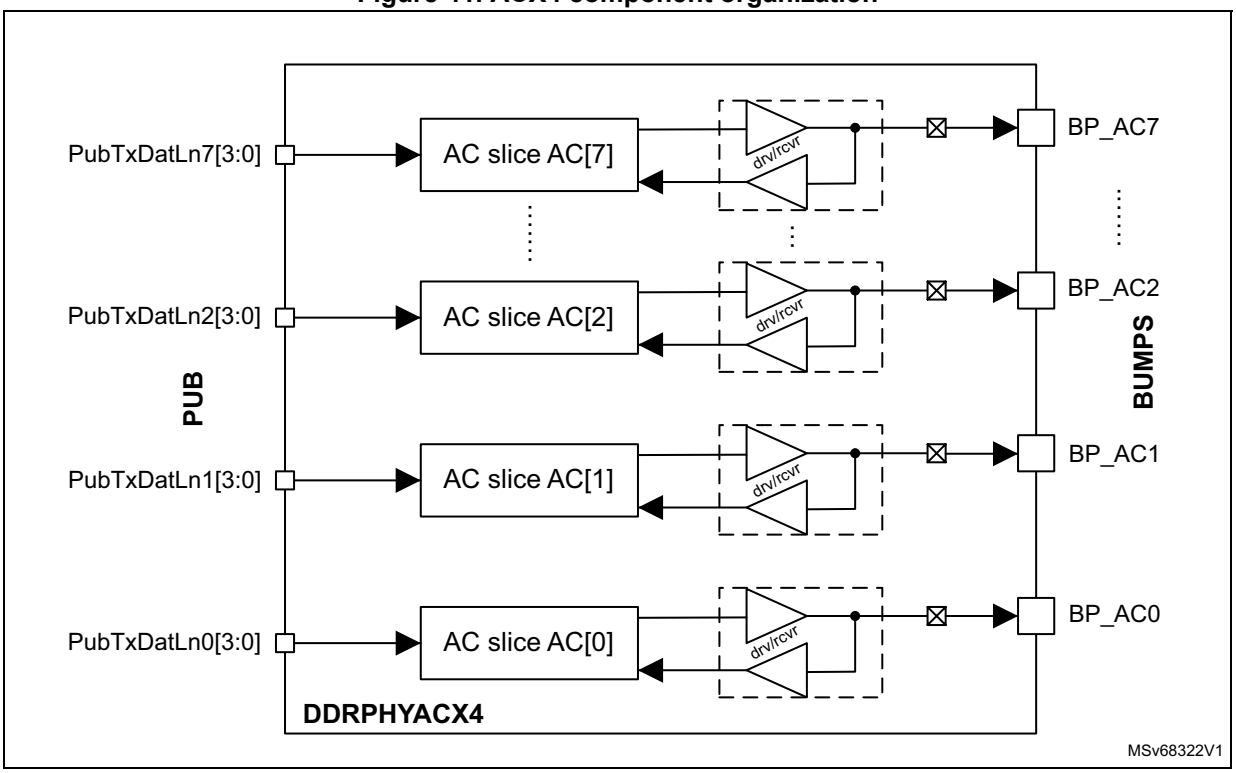
The reset takes the asynchronous reset pin, the PwrOk pin, and dfi_reset_n signal, combines them, and synchronizes to the DFI clock. This is used by ACX4 and DBYTE.

14.4.2 ACX4

The DDR SDRAM address/command PHY (ACX4) provides an address and command interface to the external SDRAM memories. *Figure 41* shows the organization of the ACX4.

The address/command (AC) consists of multiple design slices. The AC slice is used for all address and command and clk/cs/odt signals. Each slice contains all logic required for both output (address/command) and input (loopback) paths of the respective signals.
This configuration of DDRPHYC uses eight ACX4 macros

Figure 41. ACX4 component organization



14.4.3 DBYTE

The DDR SDRAM data PHY (DBYTE) provides data interface to one byte and matching strobe of an external SDRAM memory.

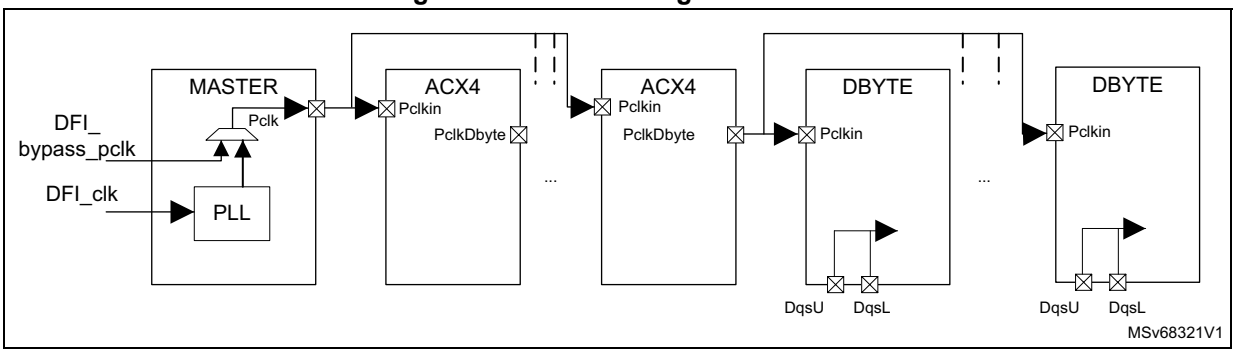
On the writing path, the controlled data pipeline logic inside the DBYTE slice converts the DFI-rate signals from the PUB to double-data rate (DDR) signals going to the SDRAM.

On the reading path, the read data from the SDRAM is written using the DQS clock into a four-deep asynchronous FIFO that is implemented inside the DQ slice, and read using the controller clock. A per-bit delay line at the input of the DQ slice is used to optionally deskew the read data with respect to the read strobe in order to maximize the read data eye.

14.4.4 PHY clocking

Figure 42 presents the PHY clocking distribution. The main input clock is the DFI clock, coming from the DDRSS PLL2 (configured through RCC dedicated registers).
In the MASTER, the embedded PLL multiplies the DFI clock by four, and is fed to ACX4 and DBYTE macros.

Figure 42. PHY clocking distribution



Only one side of the MASTER Pclk tree is represented. The MASTER feeds the clock to the ACX4 on both sides. Only the ACX4 closest to each DBYTE on each side feeds the clock to the DBYTE. All other PClkDbyte are left unconnected.

Note: For a data transfer less than 600 MT/s, the MASTER inner PLL can be bypassed. In this case, the clock coming from the DFI_Bypass_pclk is fed to the ACX4, and then to the DBYTE. A specific register in the DDRSS registers must be enabled for this purpose.

14.4.5 PUB (PHY utility block)

The PUB provides control and configuration for all the PHY hard macros (MASTER, 8x ACX4, 2x DBYTE). It handles the following functions for the PHY:

- driving initialization sequences
- providing all training functions
- implementing the DFI protocol
- managing the low power states
- implementing registers and a register access bus

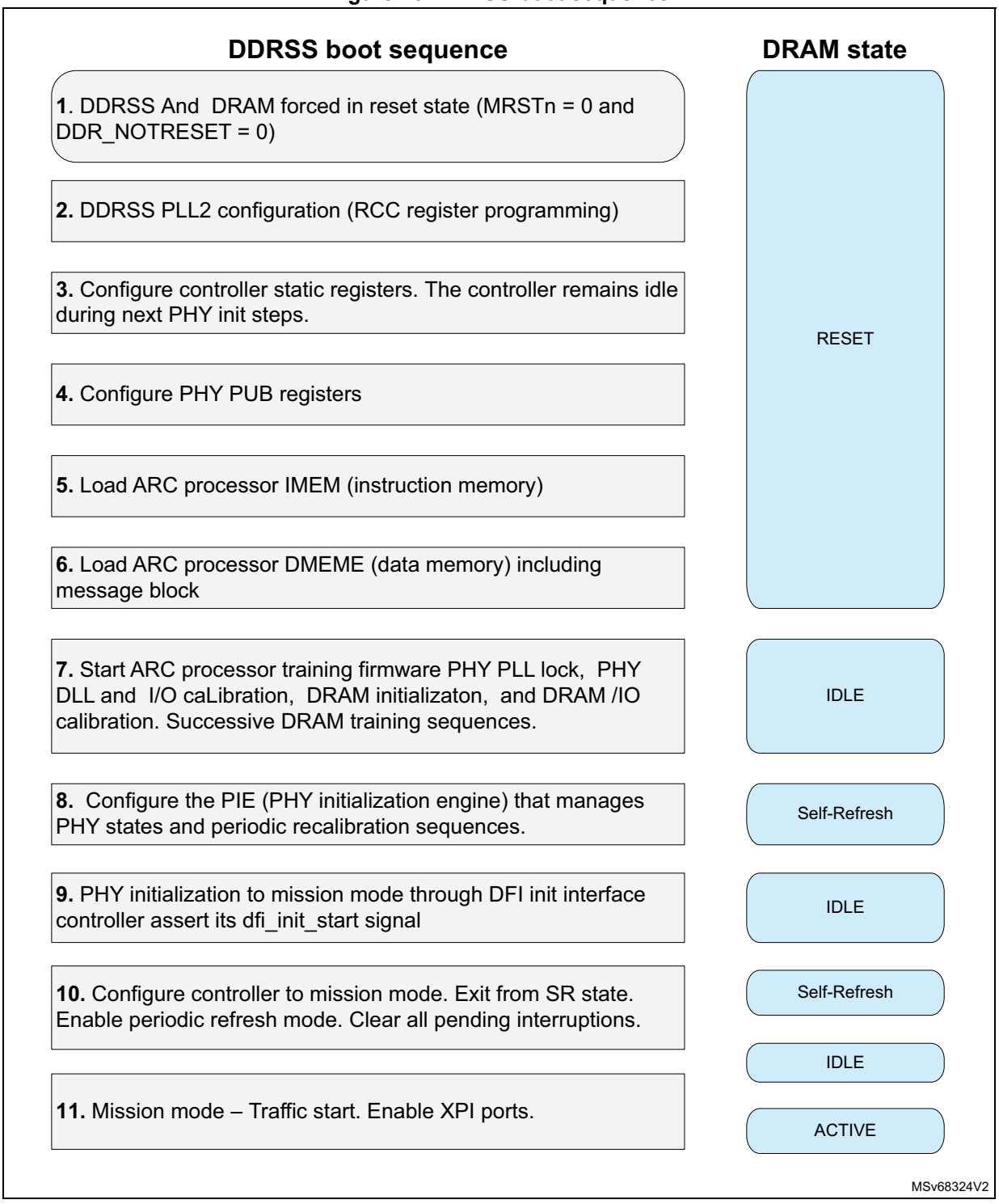
The PUB supports a DFI 4.0 interface with the DDRCTRL. Its registers are accessible through an APB interface. It also provides a JTAG interface for test purposes.

The PUB includes a programmable microcontroller unit (PMU) to support the training. Once the PHY is initialized and the firmware is loaded in the IMEM and DMEM, the PMU executes the training firmware, and is then shutdown.

14.4.6 PHY and SDRAM initialization

The figure below presents the initialization flow for the whole DDRSS.

Figure 43. DDRSS boot sequence



Details on the boot sequence are the following:

1. Bring up the power: power supplies can come up and stabilize in any order.
2. Through RCC registers, program the PLL2 to the desired frequency, and wait for the PLL to be locked. Meanwhile, DDRCTRL and DDRPHYC are kept under reset.
3. Configure the controller static registers: once the controller is configured, its XPI ports remain disabled while the DDRPHYC initialization and training is not completed.
4. Configure the PUB registers.
5. Load the PMU instruction memory: before the PHY training firmware can be run, the firmware program image must be loaded into the instruction memory SRAM. This is provided by the STM32CubeMx tool. After this stage, the PHY clock frequency (PLL2 setting through the RCC registers) must run at the desired frequency.
6. Load the PMU data memory (DMEM). In order to run the firmware, some information must be given about the system, the memory configuration, and the memory training steps to run.
7. Start the PMU training firmware: the training firmware can now be run. It configures the PHY PLL, calibrates the PHY DLL and I/Os, initializes the DRAM and calibrates DRAM I/Os, and runs the successive SDRAM training stages, depending on the SDRAM connected.
8. Once the firmware training is completed, the PMU is shutdown, and the retraining is performed by the PHY init engine. The PIE also manages the PHY state. Hence PIE registers must be written. Meanwhile the SDRAM is in SR.
9. PHY initialization to the mission mode: the controller sets the PHY into the mission mode through the DFI interface (dfi_init_start signal).
10. Configure the controller to the mission mode: gets the controller out of SR, and enables the periodic refresh. Clears all pending interruptions generated during the training (if any).
11. The DDRSS is ready to the mission mode: enables the controller AXI ports.

14.4.7 DRAM initialization sequence

This section describes the successive steps to initialize the DRAM. All training steps are common to LPDDR4, DDR4, DDR3, except the command address training (*Section 14.4.9*) which is only for LPDDR4.

The DDR initialization sequence is provided by the STM32CubeMx: the training firmware, including data and sequence is provided by the STM32CubeMx according to configuration input from the user. The explanations hereafter provide details to help understanding the initialization sequence, but the different steps of the sequence are abstracted to the user through the STM32CubeMx.

The DDRPHYC training sequence must be performed in the order described in this section. The proper operation is not guaranteed if any training stages are skipped or executed out of sequence. Each subsequent step is dependent on the DDRPHYC register values, internal states, and configurations (of both the DDRPHYC and the DRAMs) that are set during prior steps. Stopping and resuming training is not supported. The reconfiguration of the DDRPHYC or DRAM between, or during, training steps is not supported. The violation of any of these requirements may result in an inoperable memory system.

14.4.8 Device initialization

The device initialization performs the basic initialization of the DDRPHYC and DRAM that is required before any training stage is run. The initialization includes tasks such as locking the internal PLL and DLLs to the current input clock, and calculates the parameters to accelerate future lock operations, resetting the DRAMS, and writing the appropriate mode registers (MR). The training may modify the initial MRs from the values provided in the message block to avoid putting the DRAMs into an untrainable state.

14.4.9 Command address (CA) training

This step is only for LPDDR4.

During the CA training, the DRAM is put into a mode where the command/address lines are sampled and fed back on the DQ lines. A series of bit patterns are sent with varying timing to determine the valid delay range for each bit. The timing is then set to the optimal value for each bit.

This step compensates for the differences between CK signals and command/address signals to the DRAM devices.

A single delay for each ANIB is calculated by finding the time where all bits of each ANIB are sampled correctly at each device, and changing the delay of the ANIB signals to place the middle of this region as close to the edge of the clock used to capture the values.

Note: The LPDDR4 specification requires that the DRAM uses a slower frequency clock prior to training the command bus. The DDRPHYC has dedicated circuits to generate a divided down clock that is designed to meet these requirements, without the user having to change the input clock to the DDRPHYC.

14.4.10 Read gate training

The goal of read gate training (RxEn training) is to determine the round-trip latency, including RCAS latency, between the DDRPHYC and the DRAM. The round-trip latency must be known so that the DDRPHYC receivers are activated by the read gate at the right time in order to capture the full burst of read data. All DBYTE instances can be trained in parallel, but each rank must be trained independently.

This step compensates the delay for the read command to reach each DRAM, and to have the DQS pulse for the read return to the DDRPHYC. This delay does include all DRAM DQS related parameters such as t_{DQSCK}. This delay does not include the read CAS latency, which is known ahead of time.

A delay is calculated for each nibble of each DBYTE to each timing group by finding the time of the first rising edge of DQS for each timing group, and changing the delay of the RxEnb to place it as close to the middle of the DQS preamble as possible.

14.4.11 Write leveling fine phase training

The write leveling fine grain phase is performed first.

The goal of write leveling fine phase training is to align DQS rising edge with the MEMCLK rising edge at each DRAM device. The write leveling fine phase training is required to compensate for any analog phase differences between MEMCLK and DQS distribution.

This fine training ensures that the edges are aligned correctly, but not necessarily at the correct WCAS latency. The coarse part of the write leveling training calculates the delay to

ensure the edges aligned with the correct WCAS latency. See JEDEC specifications for example timing diagrams.

This step compensates for the phase difference between the CK signal and the DQS at each DRAM. A delay is calculated for each DBYTE to each timing group by using the write leveling mode of DRAMs to place the DQS rising edge as close as possible to the CK rising edge.

14.4.12 Read deskew

The read DQ deskew training compensates for the delay differences, primarily caused by board routing and DRAM DQ output skew, among the DQ lanes during reads. The read deskew must not be skipped in silicon, even on systems without much skew between lanes. The failure to run read deskew can cause part of the read-eye to be missed by later stages, leading to non-optimal results.

The read deskew training is implemented by using per lane delays to align the beginnings of the passing regions for each nibble. Because the read enable and write leveling are completed, the read deskew can send simple writes and reads to the devices to collect data on the skews, such as using MPR pattern as 0xB2B2. The firmware sets up the chosen pattern, forces the receive clock delay to 0, and then runs an in-phase left search, to find the largest passing per-lane delays for every lane. Any nibble, which lanes all succeed in finding a left edge, has all per-lane delays recorded while the failing lanes continue to try incrementally a larger receive clock delay, extending the search range, until every nibble finds a working set of per-lane delays.

This step compensates for the delay difference between DQ signals from each DRAM, and ensures the full receive-eye exists at positive rxclkdlys. A delay is calculated for each signal in each DBYTE from each timing group to align all the DQ signals with each other as close as possible.

14.4.13 Read 1D (SI friendly) training

The read 1D fine training is done so that reads can be trained to work and perform further training to properly train the write operations. The result of this training is not guaranteed to be optimal but is guaranteed to work well enough to perform the write training.

Once the write training is complete, the read DQ/DQS timing can be optimized further with longer more complex patterns to a final trained position. The firmware implements this step of training by programming a pattern into an MPR, then running an automatic eye search using the read clock delay. The results of the automatic eye search are then used to write the read clock delay values to their eye midpoints.

This step compensates for the delay difference between the DQ and the DQS signals from each DRAM device. This step also moves the sampling edge of DQS to the center of the DQ eye. A delay is calculated for the upper and lower nibble of each DBYTE separately by calculating the compound eye for all signals in each nibble and centering the sampling edge of DQS in this eye.

14.4.14 Write leveling coarse delay training

The TX DQS is phase-aligned to MEMCLK during the write leveling fine phase training (*Section 14.4.11*), but the TX DQS latency may not match the CAS write latency.
As the TX DQS delay is already phase-aligned, it can be incorrect by some integer multiple of MEMCLK periods.

During initialization, the course search calculates an educated guess for the delay between the transmitted DQS and DQ signals. This delay works for writes but is not delay margin optimized. The coarse search writes a relatively slow square wave to the rank being trained and then reads back the values. The DDRPHYC generates extra DQS pulses to ensure that DRAMs see enough DQS pulses during the time of the write even if the timing is off.

Any error in the comparisons between what is written to the DRAM and what is read out can be attributed to a coarse offset between DQS and MEMCLK signals. The firmware adds to the DQS delay currently set, and reruns the square wave write/read sequence to check if the new delay has fixed the errors. The firmware continues to loop, adding delay, until the data comes back without issue. If all reasonable delay settings fail to return the data as it was written, the firmware terminates the training. The delays start to be unreasonable around the 20 UI mark. Such a large delay likely means something has gone wrong.

This step compensates for the delay difference between the CK signal and the DQS at each DRAM. A delay is calculated for each DBYTE to each timing group by using the write leveling mode of the DRAMs to place the rising edge of DQS as close as possible to the rising edge of CK.

14.4.15 Write 1D training

At this point, the user can safely read back arbitrary data from DRAM, even if the read training is not perfectly centered on the read eye. The objective of this training is to center the write DQ timing in the write DQS eye (write DQS timing is set in the write leveling steps). The DDRPHYC has per-lane write DQ adjustment, allowing all the lanes to simultaneously train to their own value.

The write DQ/DQS delay center training is implemented by an automatic eye search that varies the delay on the write DQ signals. Once the automatic eye search has completed, the resulting eye mid-points are written to the DQ delay DDRPHYC register.

This step compensates for the delay difference between DQ and DQS signals to each DRAM. This step also moves the center of each DQ eye to sampling edge of DQS.

14.4.16 Read 1D training (optimized delay centering)

This training stage is almost the same as the read 1D (SI friendly) (*Section 14.4.13*) delay centering presented above but using longer more complex data patterns known as PRBS patterns. The complex PRBS patterns are used to obtain a more accurate estimate of the mission mode eye and to optimize the DQS position for electrically stressful patterns. Like the write DQ/DQS delay centering, all lanes are trained in parallel, but each nibble shares a common read delay setting. If read DBI is enabled the DBI lanes are trained at the same time as the DQ lanes.

This step compensates for the delay difference between DQ and DQS signals from each DRAM. This step also moves the sampling edge of DQS to the center of the DQ eye.
A delay is calculated the upper and lower nibble of each DBYTE separately by calculating the compound eye for all signals in each nibble and centering the sampling edge of DQS in this eye.

14.4.17 Max read latency training

When reading data from the DRAMs, the received data is temporarily buffered in an internal receive data FIFO. After a short period, called the read latency, the data in the FIFO is

guaranteed to be valid. The read latency training stage finds the smallest read latency that works for all FIFOS in the DDRPHYC, limiting the system memory latency.

The training is done by setting the read latency very low, then incrementing it while repeatedly performing reads. Most of these reads fail because the read latency is too small. As the read latency approaches the minimum correct value, the reads on some bytes begins to pass. Once the read passes on all bytes, the firmware stops searching, and adds an offset passed in through the message block field “DFIMRLMargin”. By default, the DDRPHYC trains all max read latency to the same value, ensuring all data are aligned on the DFI.

14.5 DDRPHYC registers

14.5.1 DDRPHYC MtestMuxSel register (DDRPHYC_ANIBx_MTESTMUXSEL)

Address offset: 0x068 + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Digital observation pin control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MTESTMUXSEL[5:0]**:

Controls for the 64-1 mux for asynchronous data to the digital observation pin.
0x0: Drives 0 from this chiplet (allows flat OR of pass-through information).
0x01 to 0x1F: Selects local data from AC, DBYTE, and MASTER macros.
0x20: Drives 0 from PUB synthesized logic path into mux.
0x21 to 0x3F: Selects local data from PUB AC, DBYTE, and MASTER synthesized logic.
Others: Reserved

14.5.2 DDRPHYC AForceDrvCont register (DDRPHYC_ANIBx_AFORCEDRVCONT)

Address offset: 0x09C + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Force address/command driven (lanes A3-A0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFORCEDRVCONT[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AFORCEDRVCONT[3:0]**:

Forces continuous drive, per-lane, of the ACX4 instance controlled by this register.
Setting this register causes the PHY to drive the target lane when dfi_init_complete = 1.
Bit [0]: Controls lane 0 of the target ACX4.
Bit [1]: Controls lane 1 of the target ACX4.
Bit [2]: Controls lane 2 of the target ACX4.
Bit [3]: Controls lane 3 of the target ACX4.

14.5.3 DDRPHYC AForceTriCont register (DDRPHYC_ANIBx_AFORCETRICONT)

Address offset: 0x0A0 + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Force address/command tristate (lanes A3-A0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFORCETRICONT[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AFORCETRICONT[3:0]**:

Forces tristate control, per-lane, of the ACX4 instance controlled by this register.
Setting this register causes the PHY to tristate the target lane when dfi_init_complete = 1.
Bit [0]: Controls lane 0 of the target ACX4.
Bit [1]: Controls lane 1 of the target ACX4.
Bit [2]: Controls lane 2 of the target ACX4.
Bit [3]: Controls lane 3 of the target ACX4.

14.5.4 DDRPHYC address Tx impedance control register (DDRPHYC_ANIBx_ATXIMPEDANCE)

Address offset: 0x10C + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 03FF

Tx impedance of address driver cells

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ADRVSTRENN[4:0]					ADRVSTRENP[4:0]				
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:5 **ADRVSTRENN[4:0]**: 5-bit bus used to select the target pull-down output impedance
00000: 120.0 Ω
00001: 60.0 Ω
00011: 40.0 Ω
00111: 30.0 Ω
01111: 24.0 Ω
11111: 20.0 Ω

Bits 4:0 **ADRVSTRENP[4:0]**: 5-bit bus used to select the target pull-up output impedance
00000: 120.0 Ω
00001: 60.0 Ω
00011: 40.0 Ω
00111: 30.0 Ω
01111: 24.0 Ω
11111: 20.0 Ω

14.5.5 DDRPHYC ATestPrbsErr register (DDRPHYC_ANIBx_ATESTPRBSERR)

Address offset: 0x14C + 0x4000 * x, (x = 0 to 7)

Reset value: 0xXXXX XXXX

Address loopback PRBS error status for an entire ACX4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTPRBSERR[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **ATESTPRBSERR[3:0]**: Overall error indicator for each PRBS bump checker
Bit[0]: Lane 0 error status (1 = errors found, 0 = no errors)
Bit[1]: Lane 1 error status
Bit[2]: Lane 2 error status
Bit[3]: Lane 3 error status

14.5.6 DDRPHYC address Tx slew rate and predriver control register (DDRPHYC_ANIBx_ATXSLEWRATE)

Address offset: 0x154 + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 07FF

Controls slew rates of address/command/memclk drivers. Tx predriver cal trims driver timing to account for clock duty-cycle variation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ATXPREDRVMODE[2:0]			ATXPREN[3:0]				ATXPREP[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **ATXPREDRVMODE[2:0]**:
Controls predrivers to adjust timing of turn-on/-off of pull-up/-down segments.
- Recommended settings if the ANIB drives CK pins:
DDR3 = 000
DDR4 = 000
LPDDR4 = 001
- Recommended settings if the ANIB does not drive CK pins:
DDR3 = 011
DDR4 = 011
LPDDR4 = 001

Bits 7:4 **ATXPREN[3:0]**: 4-bit binary trim for the driver pull-down slew rate
0000 has a slower slew rate than 1111.

Bits 3:0 **ATXPREP[3:0]**: 4-bit binary trim for the driver pull-up slew rate
0000 has a slower slew rate than 1111.

14.5.7 DDRPHYC ATestPrbsErrCnt register (DDRPHYC_ANIBx_ATESTPRBSERRCNT)

Address offset: 0x158 + 0x4000 * x, (x = 0 to 7)

Reset value: 0xXXXX XXXX

Address loopback test result

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATESTPRBSERRCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ATESTPRBSERRCNT[15:0]**: Overall error indicator for each PRBS bump checker
Bit [3:0]: Lane 0 error count
Bit [7:4]: Lane 1 error count
Bit [11:8]: Lane 2 error count
Bit [15:12]: Lane 3 error count

14.5.8 DDRPHYC ATxDly register (DDRPHYC_ANIBx_ATXDLY)

Address offset: 0x200 + 0x4000 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Address/command delay

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATXDLY[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **ATXDLY[6:0]**: Trained for LPDDR4 to generate timed address and command signals to the DRAMs (per ACX4)
For DDR3/DDR4, it is recommended that ACX4 address and commands have its ATxDly configured to 0x0. The four signals generated by an ACX4 have a common timing.
Bit[6]: Coarse delay (one unit of delay is 1 UI).
Bit[5]: Reserved.
Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32).

Note: The target ACX4 used for the memory clock generation must have its ATxDly configured to 0x0.

14.5.9 DDRPHYC DBYTE module disable register (DDRPHYC_DBYTEx_DBYTEMISCMODE)

Address offset: 0x40000 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Controls the DBYTE enable/disable state.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBYTE DISABLE	Res.	Res.
													rw		

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DBYTEDISABLE**: DBYTE disable

If DBYTE is not enabled, it receives no clocks, and remains in reset.

0: DBYTE enabled

1: DBYTE disabled

Note: This bit must only be changed at PHY initialization step C.

Bits 1:0 Reserved, must be kept at reset value.

14.5.10 DDRPHYC MtestMuxSel register (DDRPHYC_DBYTEx_MTESTMUXSEL)

Address offset: 0x40068 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Digital observation pin control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MTESTMUXSEL[5:0]**:

Controls for the 64-1 mux for asynchronous data to the digital observation pin.

0x0: Drives 0 from this chiplet (allows flat 'OR' of pass-through information).

0x01 to 0x1F: Selects local data from AC, DBYTE, and MASTER macros.

0x20: Drives 0 from PUB synthesized logic path into mux.

0x21 to 0x3F: Selects local data from PUB AC, DBYTE, and MASTER synthesized logic.

14.5.11 DDRPHYC DFIMRL register (DDRPHYC_DBYTEx_DFIMRL)

Address offset: 0x40080 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIMRL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DFIMRL[4:0]**: DFI maximum read latency (MRL)

This bit must be trained to ensure that the Rx-data FIFO is not read until after all DBYTEs have their read data valid. This gives the value (units of two memory clocks), between dfi_rddata_en and dfi_rddata_valid.

A unit change in the LSB is a change in MRL of two memory clocks.

14.5.12 DDRPHYC VrefDAC0_ry register (DDRPHYC_DBYTEx_VREFDAC0_Ry)

Address offset: 0x40100 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 8)

Reset value: 0x0000 0000

PHY Rx V_{REF} DAC control for DQ receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFDAC0[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **VREFDAC0[6:0]**: PHY Rx V_{REF} DAC control for rxdq cell internal V_{REF}

DAC control for rxdq cell internal V_{REF}, trained by firmware. V_{REF} generators have different ranges, depending on the mission mode settings for

- [DqDqsRcvCntrl:MajorMode,DqDqsRcvCntrl]: ExtVrefRange]

[000,0]: $V_{REF} = V_{DDQ} \times (0.287 + VREFDAC0[6:0] \times 0.00330)$

[000,1]: $V_{REF} = V_{DDQ} \times (0.250 + VREFDAC0[6:0] \times 0.00385)$

[011,0]: $V_{REF} = V_{DDQ} \times (0.510 + VREFDAC0[6:0] \times 0.00345)$

[011,1]: $V_{REF} = V_{DDQ} \times (0.453 + VREFDAC0[6:0] \times 0.00385)$

[010,0]: $V_{REF} = V_{DDQ} \times (0.047 + VREFDAC0[6:0] \times 0.00367)$

Others: Reserved

14.5.13 DDRPHYC data Tx impedance controls register (DDRPHYC_DBYTEx_TXIMPEDANCECTRL0_By)

Address offset: 0x40104 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 1)

Reset value: 0x0000 0FFF

Tx DQ driver impedance when equalization is enabled per nibble

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DRVSTRENDQN[5:0]						DRVSTRENDQP[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:6 **DRVSTRENDQN[5:0]**: 6-bit bus used to select the target pull-down output impedance
Note: See DrvStrenDqP for list of possible impedances.

Bits 5:0 **DRVSTRENDQP[5:0]**: 6-bit bus used to select the target pull-down output impedance
000000: HiZ
000001: 480.0
000010: 240.0
000011:160.0
001000: 120.0
001001: 96.0
001010: 80.0
001011: 68.6
011000: 60.0
011001: 53.3
011010: 48.0
011011: 43.6
111000: 40.0
111001: 36.9
111010: 34.3
111011: 32.0
111110: 30.0
111111: 28.2

14.5.14 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL_By)

Address offset: 0x4010C + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 1)

Reset value: 0x0000 05B0

Connected to the Rx DQS and Rw DQ mode selects and trim controls, per nibble

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	GAINCURRADJ[4:0]					MAJORMODEDBYTE[2:0]			DFECTRL[1:0]		EXTVREFRANGE	SELANALOGVREF
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:7 **GAINCURRADJ[4:0]**:
Adjusts gain current of RX amplifier stage. It is recommended to use default values for this bitfield.

Bits 6:4 **MAJORMODEDBYTE[2:0]**: Selection of the major mode of operation for the receiver
These settings are determined by PHY configuration based on DRAM protocol.
000: Input signal is terminated/referenced to $V_{DDQ}/2$ (or unterminated).
010: Input signal is terminated/referenced to V_{SSQ}.
011: Input signal is terminated/referenced to V_{DDQ}.
Other: Reserved

Bits 3:2 **DFECTRL[1:0]**:
00: DFE off
01: DFE on
10: Train DFE0 amplifier
11: Train DFE1 amplifier
Note: These settings are determined by PHY training firmware, and must not be overridden.

Bit 1 **EXTVREFRANGE**: Extends the range available in the local per-bit VREF generator.

Bit 0 **SELANALOGVREF**:
Setting this bit forces the local per-bit V_{REF} generator to pass the global VREFA to the samplers.

14.5.15 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL1)

Address offset: 0x40128 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0400

Various controls of Rx DQS and Rx DQ cells.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ENLPREQPDR	RXPADSTANDBYEN	POWERDOWNRCVRDQS	POWERDOWNRCVR[8:0]								
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **ENLPREQPDR**: not used

Bit 10 **RXPADSTANDBYEN**: Enables rxdq/rxdqs Standby power savings, per pad-group.

Bit 9 **POWERDOWNRCVRDQS**: Active high signal which powers down the receiver
After this bit is deasserted, the receiver cannot be used for a minimum of 100 ns.
This control powers down both upper and lower DQS receivers.
If X4TG = 0x0 in DDRPHYC_MASTER0_MASTERX4CONFIG, the upper DQS receiver is powered down.

Bits 8:0 **POWERDOWNRCVR[8:0]**: Active high signal which powers down the receiver
After this bitfield is deasserted, the receiver cannot be used for a minimum of 100 ns.
7:0 -> dq[7:0]
8 -> dbi

14.5.16 DDRPHYC DQ/DQS receiver control register (DDRPHYC_DBYTEx_DQDQSRCVCNTRL2)

Address offset: 0x40130 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Control for an aggressive power-saving feature to power down DQ and DQS receiver cells dynamically.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENRXAGRESSIVEPDR
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENRXAGRESSIVEPDR**:

14.5.17 DDRPHYC TX ODT driver strength control register (DDRPHYC_DBYTEx_TXODTDRVSTREN_By)

Address offset: 0x40134 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 1)

Reset value: 0x0000 0000

Selects desired impedance value for Host ODT, per nibble. Must be set by user before PHY initialization. Setting for each protocol:

- DDR3: ODTSTRENP - non-zero value, ODTSTRENN - same as ODTSTRENP
- DDR4: ODTSTRENP - non-zero value, ODTSTRENN - high-impedance
- LPDDR4: ODTSTRENP - high-impedance, ODTSTRENN - non-zero value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ODTSTRENN[5:0]						ODTSTRENP[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:6 **ODTSTRENN[5:0]**: ODT pull-down impedance selection
[ODTSTREN120N[2:0], ODTSTREN240N[1:0], ODTSTREN480N[0:0]]
000_00_0 = High impedance
000_00_1 = 480 Ω
000_01_0 = 240 Ω
000_01_1 = 160 Ω
001_00_0 = 120 Ω
001_00_1 = 96.0 Ω
001_01_0 = 80.0 Ω
001_01_1 = 68.6 Ω
011_00_0 = 60.0 Ω
011_00_1 = 53.3 Ω
011_01_0 = 48.0 Ω
011_01_1 = 43.6 Ω
111_00_0 = 40.0 Ω
111_00_1 = 36.9 Ω
111_01_0 = 34.3 Ω
111_01_1 = 32.0 Ω
111_11_0 = 30.0 Ω
111_11_1 = 28.2 Ω

Bits 5:0 **ODTSTRENP[5:0]**: ODT pull-up impedance selection
[ODTSTREN120P[2:0], ODTSTREN240P[1:0], ODTSTREN480P[0:0]]
000_00_0 = High impedance
000_00_1 = 480 Ω
000_01_0 = 240 Ω
000_01_1 = 160 Ω
001_00_0 = 120 Ω
001_00_1 = 96.0 Ω
001_01_0 = 80.0 Ω
001_01_1 = 68.6 Ω
011_00_0 = 60.0 Ω
011_00_1 = 53.3 Ω
011_01_0 = 48.0 Ω
011_01_1 = 43.6 Ω
111_00_0 = 40.0 Ω
111_00_1 = 36.9 Ω
111_01_0 = 34.3 Ω
111_01_1 = 32.0 Ω
111_11_0 = 30.0 Ω
111_11_1 = 28.2 Ω

14.5.18 DDRPHYC status of RX FIFO consistency checks register (DDRPHYC_DBYTEx_RXFIFOCHECKSTATUS)

Address offset: 0x40158 + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

When a PhyRxFifoCheck interrupt is posted, these bits indicate which pair of FIFO pointers do not match as expected. The non-zero pointer values can be read from DDRPHYC_DBYTEx_RXFIFOCHECKERRVALUES. Setting PHYRXFIFOCHECKCLR in DDRPHYC_MASTER0_PHYINTERRUPTCLEAR clears these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOLOCUERR	RXFIFOLOCERR
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RXFIFOLOCUERR**:
If this bit is set, the read pointer (DFI side) on the read FIFO associated with data bits [7:4] did not match. The write pointer (DQS side) on the read FIFO associated with data bits [7:4].

Bit 0 **RXFIFOLOCERR**:
If this bit is set, the read pointer (DFI side) on the read FIFO associated with data bits [3:0] did not match. The write pointer (DQS side) on the read FIFO associated with data bits [3:0].

14.5.19 DDRPHYC captured values associated with an RxFIFO consistency error register (DDRPHYC_DBYTEx_RXFIFOCHECKERRVALUES)

Address offset: 0x4015C + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Contains the value of the FIFO pointer for the first mismatch captured in DDRPHYC_DBYTEx_RXFIFOCHECKSTATUS. Setting PHYRXFIFOCHECKCLR in DDRPHYC_MASTER0_PHYINTERRUPTCLEAR clears these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOWRLOCUERRVALUE[3:0]				RXFIFORDLOCUERRVALUE[3:0]				RXFIFOWRLOCERRVALUE[3:0]				RXFIFORDLOCERRVALUE[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **RXFIFOWRLOCUERRVALUE[3:0]**: First error value captured for the write pointer (DQS side) on the read FIFO associated with data bits [7:4]

Bits 11:8 **RXFIFORDLOCUERRVALUE[3:0]**: First error value captured for the read pointer (DFI side) on the read FIFO associated with data bits [7:4]

Bits 7:4 **RXFIFOWRLOCERRVALUE[3:0]**: First error value captured for the write pointer (DQS side) on the read FIFO associated with data bits [3:0]

Bits 3:0 **RXFIFORDLOCERRVALUE[3:0]**: First error value captured for the read pointer (DFI side) on the read FIFO associated with data bits [3:0]

14.5.20 DDRPHYC data receive FIFO pointer values register (DDRPHYC_DBYTEx_RXFIFOINFO)

Address offset: 0x40160 + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Provides visibility of the read and write pointers of the Rx FIFOs for debug purposes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOWRLOCU[3:0]				RXFIFORDLOCU[3:0]				RXFIFOWRLOC[3:0]				RXFIFORDLOC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **RXFIFOWRLOCU[3:0]**: Mission mode write pointer of the upper-nibble Rx FIFO

Bits 11:8 **RXFIFORDLOCU[3:0]**: Mission mode read pointer of the upper-nibble Rx FIFO

Bits 7:4 **RXFIFOWRLOC[3:0]**: Mission mode write pointer of the lower-nibble Rx FIFO

Bits 3:0 **RXFIFORDLOC[3:0]**: Mission mode read pointer of the lower-nibble Rx FIFO

14.5.21 DDRPHYC Rx FIFO visibility register (DDRPHYC_DBYTEx_RXFIFOVISIBILITY)

Address offset: 0x40164 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Allows the contents of the data Rx FIFO to be read via RXFIFOCONTENTSDBI, RXFIFOCONTENTSDQ7654, RXFIFOCONTENTSDQ3210 bitfields. The FIFO must be quiescent (no actual operations in progress, while reading the FIFO): read from the Rx FIFO must be after the max-read-latency (DFIMRL) time.

Procedure to read the contents of the entire Rx FIFO:

1. Wait for any outstanding reads to complete (min 2*DFIMRL MemClks).
2. Read and store RXFIFOWRLOC[3:0] and RXFIFOWRLOCU[3:0] (must be equal).
3. Perform four BL8 memory reads.
4. Wait at least 2 * DFIMRL MemClks.
5. For (INIB = 0, INIB < 8, INIB++) - [
   a) MYRDPTR[2:0] = (stored RXFIFOWRLOC[3:0]>>1) + INIB (can wrap 7 to 0)
   b) Write addr = RXFIFOVISIBILITY, data = [RXFIFORDEN = 0, RXFIFORDPTROVR = 1, RXFIFORDPTR = myRdPtr]
   c) Write addr = RXFIFOVISIBILITY, data = [RXFIFORDEN = 1, RXFIFORDPTROVR = 1,RXFIFORDPTR = myRdPtr].
   d) Write addr = RXFIFOVISIBILITY, data = [RXFIFORDEN = 0, RXFIFORDPTROVR = 1, RXFIFORDPTR = myRdPtr].
   e) Wait for 8 DfiClk.
   f) Read addr = RXFIFOCONTENTSDBI.
   g) Read addr = RXFIFOCONTENTSDQ7654.
   h) Read addr = RXFIFOCONTENTSDQ3210.
   i) Unpack/align resulting data as described in descriptions of RXFIFOCONTENTS registers, and map according to the configuration of Dq<7..0>LnSel[2:0]. -].
6. Evaluate/compare with data that travels through normal read path (dfi_read_data_dbi,dfi_read_data).
7. Write [RXFIFORDEN = 0, RXFIFORDPTROVR = 0, RXFIFORDPTR = 0].
8. PHY is now available for normal mission-mode operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFORDEN	RXFIFORDPTROVR	RXFIFORDPTR[2:0]		
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RXFIFORDEN**:

Pulse [set 0-->1-->0] this bit to capture the FIFO contents.

Bit 3 **RXFIFORDPTROVR**:

This bit is programmable as follows:

0: Normal operation - Mission mode read pointer is enabled.

1: Override - Control of the Rx FIFO read pointer is ceded to this bit.

Bits 2:0 **RXFIFORDPTR[2:0]**:

If RXFIFORDPTROVR is set, this bitfield selects the Rx FIFO entry. This bitfield addresses 4-bit units of the 8x4b (32 entry) FIFO:
rdfifo_nibble_address[2:0] = RXFIFORDPTR[2:0].
For example, RXFIFORDPTR[2:0] = 2 enables reading bit-entries 11..8.
The exact location of read data depends on the prior history of reads and PHY initialization.

14.5.22 DDRPHYC Rx FIFO content DQ321x register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ3210)

Address offset: 0x40168 + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Rx FIFO contents, lane[3:0].

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOCONTENTSDQ3210[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFIFOCONTENTSDQ3210[15:0]**: Window into the contents of the Rx FIFO, as controlled by RXFIFOVISIBILITY.

This bitfield reads 4 bits at a time from lane0..3, from the four FIFO entries addressed by rdfifo_nibble_address[2:0] = RXFIFORDPTR[2:0].
[15:12] = [lane3_ui3,lane3_ui2,lane3_ui1,lane3_ui0] where ui0 is the first occurring bit
[11: 8] = [lane2_ui3,lane2_ui2,lane2_ui1,lane2_ui0]
[7: 4] = [lane1_ui3,lane1_ui2,lane1_ui1,lane1_ui0]
[3: 0] = [lane0_ui3,lane0_ui2,lane0_ui1,lane0_ui0]

Note: The DBYTE lane of a given index is not the same as a memory DQ of the same index unless the Dq<7..0>LnSel[2:0] have their default/reset value.

14.5.23 DDRPHYC Rx FIFO content DQ7654 register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ7654)

Address offset: 0x4016C + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Rx FIFO contents, lane[7:4].

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOCONTENTSDQ7654[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFIFOCONTENTSDQ7654[15:0]**: Window into the contents of the Rx FIFO as controlled by RXFIFOVISIBILITY.

This bitfield reads 4 bits at a time from lane4..7 from the four FIFO entries addressed by rdfifo_nibble_address[2:0] = RXFIFORDPTR[2:0].
[15:12] = [lane7_ui3,lane7_ui2,lane7_ui1,lane7_ui0]
[11: 8] = [lane6_ui3,lane6_ui2,lane6_ui1,lane6_ui0]
[7: 4] = [lane5_ui3,lane5_ui2,lane5_ui1,lane5_ui0]
[3: 0] = [lane4_ui3,lane4_ui2,lane4_ui1,lane4_ui0]

Note: The DBYTE lane of a given index is not the same as a memory DQ of the same index unless the Dq<7..0>LnSel[2:0] have their default/reset value.

14.5.24 DDRPHYC Rx FIFO content DBI register (DDRPHYC_DBYTEx_RXFIFOCONTENTSDBI)

Address offset: 0x40170 + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Rx FIFO contents, DBI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOCONTENTSDBI[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **RXFIFOCONTENTSDBI[3:0]**: Window into the contents of the Rx FIFO, as controlled by RXFIFOVISIBILITY.

This bitfield reads 4 bits at a time from DBI, from the four FIFO entries addressed by rdfifo_nibble_address[2:0] = RXFIFORDPTR[2:0].
[3: 0] = [dbi_ui3,dbi_ui2,dbi_ui1,dbi_ui0]

Note: The DBYTE DBI lane is the same as the memory DBI; it is not subject to mapping using Dq<7..0>LnSel[2:0] as the DQ are.

14.5.25 DDRPHYC Tx slew rate controls register (DDRPHYC_DBYTEx_TXSLEWRATE_By)

Address offset: 0x4017C + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 1)

Reset value: 0x0000 07FF

Controls slew rates of DQ/DQS drivers per nibble. Tx predriver cal trims driver timing to account for clock duty cycle variation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXPREDRVMODE[2:0]			TXPREN[3:0]				TXPREP[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **TXPREDRVMODE[2:0]**:
Controls predrivers to adjust timing of turn-on and turn-off of pull-up and pull-down segments.
Recommended settings:
- DDR3 = 011
- DDR4 = 010
- LPDDR4 = 001

Bits 7:4 **TXPREN[3:0]**: 4-bit binary trim for the driver pull -own slew rate
0000 has a slower slew rate than 1111.

Bits 3:0 **TXPREP[3:0]**: 4-bit binary trim for the driver pull-up slew rate
0000 has a slower slew rate than 1111.

14.5.26 DDRPHYC RxPBDlyTgx_r0 register (DDRPHYC_DBYTEx_RXPBDLYTG0_Ry)

Address offset: 0x401A0 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y=0 to 8)

Reset value: 0x0000 0000

Read DQ per-bit BDL delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBDLYTG0_R0[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **RXPBDLYTG0_R0[6:0]**: Read DQ per-bit BDL delay (timing group 0)

Trained to deskew the read DQ data that are sampled by the incoming per-nibble read DQS, to improve the composite eye. The unit of change is an LSB in the control of the delay circuit, and is not denominated in units of UI.

Note: Timing group 0 = RANK0.

14.5.27 DDRPHYC RxPBDlyTg1_rx register (DDRPHYC_DBYTEx_RXPBDLYTG1_Ry)

Address offset: 0x401A4 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 8)

Reset value: 0x0000 0000

Read DQ per-bit BDL delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBDLYTG1_R0[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **RXPBDLYTG1_R0[6:0]**: Read DQ per-bit BDL delay (timing group 1)

Note: Trained to deskew the read DQ data that are sampled by the incoming per-nibble read DQS, to improve the composite eye. The unit of change is a LSB in the control of the delay circuit, and is not denominated in units of UI.*Timing group 1 = RANK1.*

14.5.28 DDRPHYC RxEnDlyTgx_u0 register (DDRPHYC_DBYTEx_RXENDLYTG0_U0)

Address offset: 0x40200 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Trained receive enable delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RXENDLYTG0_U0[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **RXENDLYTG0_U0[10:0]**: Trained receive enable delay (timing group 0)
Trained to set the delay from the memory-read command to the signal enabling the read DQS to generate read-data strobes. The trained read DQS enable asserts during the read DQS preamble and deasserts at the end of the read burst.
- Bit[10:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: Timing group 0 = RANK0.

14.5.29 DDRPHYC RxEnDlyTg1_ux register (DDRPHYC_DBYTEx_RXENDLYTG1_U0)

Address offset: 0x40204 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Trained receive enable delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RXENDLYTG1_U0[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **RXENDLYTG1_U0[10:0]**: Trained receive enable delay (timing group 1)
Trained to set the delay from the memory-read command to the signal enabling the read DQS to generate read-data strobes. The trained read DQS enable asserts during the read DQS preamble and deasserts at the end of the read burst.
- Bit[10:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: Timing group 1 = RANK1.

14.5.30 DDRPHYC RxClkDlyTgx_u0 register (DDRPHYC_DBYTEx_RXCLKDLYTG0_U0)

Address offset: 0x40230 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS to RxClk delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG0_U0[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKDLYTG0_U0[5:0]**: Trained read DQS to RxClk delay (timing group 0)
Trained to generate read DQ receive clock from the received read DQS. RxClk Delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved
There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 0 = RANK0.

14.5.31 DDRPHYC RxClkDlyTg1_ux register (DDRPHYC_DBYTEx_RXCLKDLYTG1_U0)

Address offset: 0x40234 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS to RxClk delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG1_U0[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKDLYTG1_U0[5:0]**: Trained read DQS to RxClk delay (timing group 1)
Trained to generate read DQ receive clock from the received read DQS. RxClk Delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved
There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 1 = RANK1.

14.5.32 DDRPHYC RxClkcDlyTgx_u0 register (DDRPHYC_DBYTEx_RXCLKCDLYTG0_U0)

Address offset: 0x40240 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS_c to RxClkc delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG0_U0[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKCDLYTG0_U0[5:0]**: Trained read DQS_c to RxClkc delay (timing group 0)
Trained to generate read DQ receive clock from the received read DQS_c. RxClkc delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved
There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 0 = RANK0.

14.5.33 DDRPHYC RxClkcDlyTg1_ux register (DDRPHYC_DBYTEx_RXCLKCDLYTG1_U0)

Address offset: 0x40244 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS_c to RxClkc delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG1_U0[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKCDLYTG1_U0[5:0]**: Trained read DQS_c to RxClkc delay (timing group 1)
Trained to generate read DQ receive clock from the received read DQS_c. RxClkc delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved
There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 1 = RANK1.

14.5.34 DDRPHYC DqLnSel register (DDRPHYC_DBYTEx_DQyLNSEL)

Address offset: 0x40280 + 0x4000 * x + 0x4 * y, (x = 0 to 1; y = 0 to 7)

Reset value: 0x0000 0000, 0x0000 0001, 0x0000 0002, 0x0000 0003, 0x0000 0004, 0x0000 0005, 0x0000 0006, 0x0000 0007

Maps PHY DQ lane to memory DQy.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DQYLNSEL[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **DQYLNSEL[2:0]**:

Supports mapping of PHY DQ to DRAM DQ within a byte (swizzle). Intended to undo the swizzle of board-level phy-to-dram connections such that MRR operations of binary counters return correct values.

This bitfield is per-DBYTE such that the swizzle may be different per DBYTE.

Each bit in a byte set of DqyLnSel must have a unique value within the set.

For example, if, on a DBYTE, PHY lane 3 is connected to memory DQY (on this DBYTE), then DQYLNSEL[2:0] for this DBYTE must be 3.

14.5.35 DDRPHYC TxDqDlyTgx_r0 register (DDRPHYC_DBYTEx_TXDQDLYTG0_Ry)

Address offset: 0x40300 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 8)

Reset value: 0x0000 0010

Write DQ delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQDLYTG0_R0[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TXDQDLYTG0_R0[8:0]**: Write DQ delay (timing group 0)

Trained to center the delay between the write DQ to the write DQS. Tx DQ delay (9 bits, includes 3-bit coarse and 6-bit fine delays).

- Bit[8:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine (fractional UI) delay (one unit of delay is one-thirty second of a UI = UI/32)

Note: - Bit[5]: Reserved*Timing group 0 = RANK0.*

14.5.36 DDRPHYC TxDqDlyTg1_rx register (DDRPHYC_DBYTEx_TXDQDLYTG1_Ry)

Address offset: 0x40304 + 0x4000 * x + 0x400 * y, (x = 0 to 1; y = 0 to 8)

Reset value: 0x0000 0010

Write DQ delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQDLYTG1_R0[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TXDQDLYTG1_R0[8:0]**: Write DQ delay (timing group 1)
Trained to center the delay between the write DQ to the write DQS. Tx DQ Delay (9 bits, includes 3-bit coarse and 6-bit fine delays).
- Bit1[8:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine (fractional UI) delay (one unit of delay is one-thirty second of a UI = UI/32)

Note: - Bit[5]: Reserved Timing group 1 = RANK1.

14.5.37 DDRPHYC TxDqsDlyTgx_u0 register (DDRPHYC_DBYTEx_TXDQSDLYTG0_U0)

Address offset: 0x40340 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Write DQS delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG0_U0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXDQSDLYTG0_U0[9:0]**: Write DQS delay (timing group 0)
Trained to set the delay from the memory-write command to the signal driving the write DQS. Tx DQS delay (10 bits, includes coarse and fine delays).
- Bit[9:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: This register is per nibble, per timing-group. Timing group 0 = RANK0.

14.5.38 DDRPHYC TxDqsDlyTg1_ux register (DDRPHYC_DBYTEx_TXDQSDLYTG1_U0)

Address offset: 0x40344 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Write DQS delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG1_U0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXDQSDLYTG1_U0[9:0]**: Write DQS delay (timing group 1)
Trained to set the delay from the memory-write command to the signal driving the write DQS. Tx DQS delay (10 bits, includes coarse and fine delays).
- Bit[9:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: This register is per nibble, per timing-group. Timing group 1 = RANK1.

14.5.39 DDRPHYC debug status of the DBYTE LCDL register (DDRPHYC_DBYTEx_DXLCDLSTATUS)

Address offset: 0x40390 + 0x4000 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

Status of DBYTE macro LCDL selected by LcdlStatusSel register, that is per DBYTE.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DXLCDLLIVELOCK	DXLCDLSTICKYUNLOCK	DXLCDLSTICKYLOCK	DXLCDLPHDSNAPVAL	DXLCDLFINESNAPVAL[9:0]									
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DXLCDLLIVELOCK**: Present value of whether the LCDL is locked
Valid when LCDLTSTENABLE = 1.

Bit 12 **DXLCDLSTICKYUNLOCK**: Latched value of whether the LCDL ever lost lock after the assertion of LCDLTSTENABLE

Bit 11 **DXLCDLSTICKYLOCK**: Latched value of whether the LCDL ever achieved lock after the assertion of LCDLTSTENABLE

Bit 10 **DXLCDLPHDSNAPVAL**: Value of the LCDL phase-detector output, latched by pulse on LCDLFINESNAP while LCDLTSTENABLE = 1

Bits 9:0 **DXLCDLFINESNAPVAL[9:0]**: Value of the LCDL 1UI estimate code, latched by pulse on LCDLFINESNAP while LCDLTSTENABLE = 1
Index 9 is reserved for growth.

14.5.40 DDRPHYC RxEnDlyTgx_u1 register (DDRPHYC_DBYTEx_RXENDLYTG0_U1)

Address offset: 0x40600 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Trained receive enable delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RXENDLYTG0_U1[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **RXENDLYTG0_U1[10:0]**: Trained receive enable delay (timing group 0)

Trained to set the delay from the memory-read command to the signal enabling the read DQS to generate read-data strobes. The trained read DQS enable asserts during the read DQS preamble and deasserts at the end of the read burst.

- Bit[10:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: Timing group 0 = RANK0.

14.5.41 DDRPHYC RxEnDlyTg1_u1 register (DDRPHYC_DBYTEx_RXENDLYTG1_U1)

Address offset: 0x40604 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Trained receive enable delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RXENDLYTG1_U1[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **RXENDLYTG1_U1[10:0]**: Trained receive enable delay (timing group 1)

Trained to set the delay from the memory-read command to the signal enabling the read DQS to generate read-data strobes. The trained read DQS enable asserts during the read DQS preamble and deasserts at the end of the read burst.

- Bit[10:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]:Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: Timing group 1 = RANK1.

14.5.42 DDRPHYC RxClkDlyTgx_u1 register (DDRPHYC_DBYTEx_RXCLKDLYTG0_U1)

Address offset: 0x40630 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS to RxClk delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG0_U1[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKDLYTG0_U1[5:0]**: Trained read DQS to RxClk delay (timing group 0)
Trained to generate read DQ receive clock from the received read DQS. RxClk delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 0 = RANK0.

14.5.43 DDRPHYC RxClkDlyTg1_u1 register (DDRPHYC_DBYTEx_RXCLKDLYTG1_U1)

Address offset: 0x40634 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS to RxClk delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG1_U1[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKDLYTG1_U1[5:0]**: Trained read DQS to RxClk delay (timing group 1)
Trained to generate read DQ receive clock from the received read DQS. RxClk delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

There is no coarse delay field: the max configurable delay is 31/32 UI.

Note: Timing group 1 = RANK1.

14.5.44 DDRPHYC RxClkcDlyTgx_u1 register (DDRPHYC_DBYTEx_RXCLKCDLYTG0_U1)

Address offset: 0x40640 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS_c to RxClkc delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG0_U1[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKCDLYTG0_U1[5:0]**: Trained read DQS_c to RxClkc delay (timing group 0)
Trained to generate read DQ receive clock from the received read DQS_c. RxClkc delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

There is no coarse delay field (the max configurable delay is 31/32 UI).

Note: Timing group 0 = RANK0.

14.5.45 DDRPHYC RxClkcDlyTg1_u1 register (DDRPHYC_DBYTEx_RXCLKCDLYTG1_U1)

Address offset: 0x40644 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0010

Trained read DQS_c to RxClkc delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG1_U1[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RXCLKCDLYTG1_U1[5:0]**: Trained read DQS_c to RxClkc delay (timing group 1)
Trained to generate read DQ receive clock from the received read DQS_c. RxClkc delay (6 bits, fine delay only).
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

There is no coarse delay field (the max configurable delay is 31/32 UI).

Note: Timing group 1 = RANK1.

14.5.46 DDRPHYC TxDqsDlyTgx_u1 register (DDRPHYC_DBYTEx_TXDQSDLYTG0_U1)

Address offset: 0x40740 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Write DQS delay (timing group 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG0_U1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXDQSDLYTG0_U1[9:0]**: Write DQS delay (timing group 0)

Trained to set the delay from the memory-write command to the signal driving the write DQS.
Tx DQS delay (10 bits, includes coarse and fine delays)
- Bit[9:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: This register is per nibble, per timing-group. Timing group 0 = RANK0.

14.5.47 DDRPHYC TxDqsDlyTg1_u1 register (DDRPHYC_DBYTEx_TXDQSDLYTG1_U1)

Address offset: 0x40744 + 0x4000 * x, (x = 0 to 1)

Reset value: 0x0000 0100

Write DQS delay (timing group 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG1_U1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXDQSDLYTG1_U1[9:0]**: Write DQS delay (timing group 1)

Trained to set the delay from the memory-write command to the signal driving the write DQS.
Tx DQS delay (10 bits, includes coarse and fine delays).
- Bit[9:6]: Coarse delay (one unit of delay is 1 UI)
- Bit[4:0]: Fine delay (one unit of delay is one-thirty second of a UI = UI/32)
- Bit[5]: Reserved

Note: This register is per nibble, per timing-group.Timing group 1 = RANK1.

14.5.48 DDRPHYC Rx FIFO pointer initialization control register (DDRPHYC_MASTER0_RXFIFOINIT)

Address offset: 0x80000

Reset value: 0x0000 0000

This register enables overrides of the PHY RXDATAFIFO pointers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INHIBITRXFIFORD	RXFIFOINITPTR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **INHIBITRXFIFORD**: Reserved for training firmware use
Setting this inhibits reads of the PHYRXDATAFIFO.

Bit 0 **RXFIFOINITPTR**:
Setting this bit resets the PHY RXDATAFIFO read and write pointers. This bit must be cleared to return to normal operation. This is not required in normal operation, but can be used instead of a dfi_ctrlupdate (for controllers that do not support ctrlupdate).

14.5.49 DDRPHYC ForceClkDisable register (DDRPHYC_MASTER0_FORCECLKDISABLE)

Address offset: 0x80004

Reset value: 0x0000 0000

Clock gating control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCECLKDISABLE[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **FORCECLKDISABLE[3:0]**:
This bitfield forces the gating of MEMCLKs driven from the PHY.
- Bit[0]: Controls CLK_[H/L0].
- Bit[1]: Controls CLK_[H/L]1 (if present).
- Bit[2]: Controls CLK_[H/L]2 (if present).
- Bit[3]: Controls CLK_[H/L]3 (if present).

14.5.50 DDRPHYC ForceInternalUpdate register (DDRPHYC_MASTER0_FORCEINTERNALUPDATE)

Address offset: 0x8000C

Reset value: 0x0000 0000

This register used by training firmware to force an internal PHY update event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCEINTERNALUPDATE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FORCEINTERNALUPDATE**: Used by training firmware to force an internal PHY update event
Optionally can be used by system software to issue a PHY update event. This bit can be written to 0x1 only when all of the following are true:
- The DFI interface is offline.
- The training hardware sends only DES and all transactions have retired.

This bit must stay asserted for at least 32 DFICLKs before clearing. Prematurely clearing this bit may result in an incomplete update.

14.5.51 DDRPHYC read-only displays PHY configuration register (DDRPHYC_MASTER0_PHYCONFIG)

Address offset: 0x80010

Reset value: 0xXXXX X128

Displays PHY configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PHYCONFIGDFI[1:0]		PHYCONFIGDBYTES[3:0]				PHYCONFIGANIBS[3:0]			
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **PHYCONFIGDFI[1:0]**:
Returns the following value depending on the define:
- 0x1 if DWC_DDRPHY_DFI1_EXISTS is not defined.
- 0x2 if DWC_DDRPHY_DFI1_EXISTS is defined.

Bits 7:4 **PHYCONFIGDBYTES[3:0]**:

Returns the following value depending on the define:
- 0x1 if DWC_DDRPHY_NUM_DBYTES_1
- 0x2 if DWC_DDRPHY_NUM_DBYTES_2
- 0x3 if DWC_DDRPHY_NUM_DBYTES_3
- 0x4 if DWC_DDRPHY_NUM_DBYTES_4
- 0x5 if DWC_DDRPHY_NUM_DBYTES_5
- 0x6 if DWC_DDRPHY_NUM_DBYTES_6
- 0x7 if DWC_DDRPHY_NUM_DBYTES_7
- 0x8 if DWC_DDRPHY_NUM_DBYTES_8
- 0x9 if DWC_DDRPHY_NUM_DBYTES_9
- 0xA if DWC_DDRPHY_NUM_DBYTES_10

Bits 3:0 **PHYCONFIGANIBS[3:0]**:

Returns the following value depending on the define:
- 0x3 if DWC_DDRPHY_NUM_ANIBS_3
- 0x6 if DWC_DDRPHY_NUM_ANIBS_6
- 0x8 if DWC_DDRPHY_NUM_ANIBS_8
- 0xA if DWC_DDRPHY_NUM_ANIBS_10
- 0xC if DWC_DDRPHY_NUM_ANIBS_12

14.5.52 DDRPHYC PHY general configuration register (DDRPHYC_MASTER0_PGCR)

Address offset: 0x80014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKRISEFALLMODE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **RXCLKRISEFALLMODE**:

Controls independent training for RxClk_c and RxClk_t.
0: Independent training for RxClk_c and RxClk_t is disabled (default value).
1: Independent training for RxClk_c and RxClk_t is enabled.

14.5.53 DDRPHYC test bump control1 register (DDRPHYC_MASTER0_TESTBUMPCNTRL1)

Address offset: 0x8001C

Reset value: 0x0000 0B03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TESTPOWERGATEEN	TESTEXTVREFRANGE	TESTSELEXTERNALVREF	TESTGAINCURRADJ[4:0]					TESTANALOGOUTCTRL[3:0]				TESTBIASBYPASSEN	TESTMAJORMODE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TESTPOWERGATEEN**: Do not use, for debug only

Bit 14 **TESTEXTVREFRANGE**:
Setting this bit extends the VREF DAC range for debug.
Must be set to zero in mission mode when MemAlert is enabled.

Bit 13 **TESTSELEXTERNALVREF**: Do not use, for debug only

Bits 12:8 **TESTGAINCURRADJ[4:0]**:
Adjust gain and current of analog observe RX amplifier stage at analog test point.
Recommended mission mode default = 0x01011.

Bits 7:4 **TESTANALOGOUTCTRL[3:0]**: Reserved and returns zero on reads.

Bit 3 **TESTBIASBYPASSEN**: Do not use, for debug only

Bits 2:0 **TESTMAJORMODE[2:0]**:
Selects the major mode of operation for the receiver. These modes are mutually exclusive.
011: Setting for DDR4 systems to receive Alert_n
Others: Reserved

14.5.54 DDRPHYC impedance calibration clock ratio register (DDRPHYC_MASTER0_CALUCLKINFO)

Address offset: 0x80020

Reset value: 0x0000 0320

Informs the impedance calibration engine about frequency of DfiClk.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CALUCLKTICKSPER1US[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **CALUCLKTICKSPER1US[10:0]**:
Must be programmed to the number of DfiClks in 1 µs (rounded up), with minimum value of 24.
- if (DfiClk < 24MHz) CalUclkInfo = 24
- else CalUclkInfo = (number of DfiClks in 1 µs)

14.5.55 DDRPHYC test bump control register (DDRPHYC_MASTER0_TESTBUMPCNTRL)

Address offset: 0x80028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FORCEMTESTONALERT	TESTBUMPDATASEL[5:0]						TESTBUMPTOGGLE	TESTBUMPEN[1:0]	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FORCEMTESTONALERT**:
When set, this bit causes the digital observation output pin to be driven onto BP_ALERT_N.

Bits 8:3 **TESTBUMPDATASEL[5:0]**: RVSD

Bit 2 **TESTBUMPTOGGLE**:
Controls the output function of the signal. Digital observation pin, if available in the PHY configuration.
0: The output is tristate.
1: The output is selected by MtestMuxSel.

Bits 1:0 **TESTBUMPEN[1:0]**:
Controls the output function of the signal BP_ALERT_N.
TestBumpEn Value on [BP_ALERT_N] | output
00: High-z
01: Pllout-divided-by-2, works up to full data rate.
10: PllDigTst[1], works up to 400 MHz.
11: PllDigTst[1], works up to 400 MHz.

14.5.56 DDRPHYC Seq0BDLY0 register (DDRPHYC_MASTER0_SEQ0BDLY0)

Address offset: 0x8002C

Reset value: 0x0000 0000

PHY initialization engine (PIE) delay register 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ0BDLY0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEQ0BDLY0[15:0]**: PIE delay register 0

This bitfield is available for selection by the NOP and WAIT instructions in the PIE for the delay value. Programmed by dwc_ddrphy_phyinit_I_loadPIEImage() to support frequency changes.

Programming bit[3,2,1,0]:

- These bits are used to control various delays during PLL, DLL initialization.
- These bits must be programmed with the corresponding number of MEMCLK cycles representing a fixed amount of delay irrespective of DfiClk frequency.
- When calculating the number of cycles to be programmed, it must be noted that the sequencer multiplies programmed values by a factor of eight, and values are round down to the closest integer.

Bit: Required delay | Programmed value

- Bit[0]: 0.5 µs | (0.5 µs / MEMCLK period) / 8)
- Bit[1]: 1.0 µs | (1.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | (10.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | 64/4 if (MEMCLK freq ≤ 400 MHz)
- Bit[3]: - | 132/4 if (400 MHz freq < MEMCLK freq ≤ 533 MHz)
- Bit[3]: - | 176/4 if (MEMCLK freq > 533 MHz)

14.5.57 DDRPHYC Seq0BDLY1 register (DDRPHYC_MASTER0_SEQ0BDLY1)

Address offset: 0x80030

Reset value: 0x0000 0000

PIE delay register 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ0BDLY1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEQ0BDLY1[15:0]**: PIE delay register 1

This bitfield is available for selection by the NOP and WAIT instructions in the PIE for the delay value. Programmed by dwc_ddrphy_phyinit_I_loadPIEImage() to support frequency changes.

Programming bit[3,2,1,0]:

- These bits are used to control various delays during PLL, DLL initialization.
- These bits must be programmed with the corresponding number of MEMCLK cycles representing a fixed amount of delay irrespective of DfiClk frequency.
- When calculating the number of cycles to be programmed, it must be noted that the sequencer multiplies programmed values by a factor of eight and values are round down to the closest integer.

Bit: Required delay | Programmed value

- Bit[0]: 0.5 µs | (0.5 µs / MEMCLK period) / 8)
- Bit[1]: 1.0 µs | (1.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | (10.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | 64/4 if (MEMCLK freq ≤ 400 MHz)
- Bit[3]: - | 132/4 if (400 MHz freq < MEMCLK freq ≤ 533 MHz)
- Bit[3]: - | 176/4 if (MEMCLK freq > 533 MHz)

14.5.58 DDRPHYC Seq0BDLY2 register (DDRPHYC_MASTER0_SEQ0BDLY2)

Address offset: 0x80034

Reset value: 0x0000 0000

PIE delay register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ0BDLY2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEQ0BDLY2[15:0]**: PIE delay register 2

This bitfield is available for selection by the NOP and WAIT instructions in the PIE for the delay value. Programmed by dwc_ddrphy_phyinit_I_loadPIEImage() to support frequency changes.

Programming bit[3,2,1,0]:

- These bits are used to control various delays during PLL, DLL initialization.
- These bits must be programmed with the corresponding number of MEMCLK cycles representing a fixed amount of delay irrespective of DfiClk frequency.
- When calculating the number of cycles to be programmed, it must be noted that the sequencer multiplies programmed values by a factor of 8 and values are round down to the closest integer.

Bit: Required delay | Programmed value

- Bit[0]: 0.5 µs | (0.5 µs / MEMCLK period) / 8)
- Bit[1]: 1.0 µs | (1.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | (10.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | 64/4 if (MEMCLK freq ≤ 400 MHz)
- Bit[3]: - | 132/4 if (400 MHz freq < MEMCLK freq ≤ 533 MHz)
- Bit[3]: - | 176/4 if (MEMCLK freq > 533 MHz)

14.5.59 DDRPHYC Seq0BDLY3 register (DDRPHYC_MASTER0_SEQ0BDLY3)

Address offset: 0x80038

Reset value: 0x0000 0000

PIE delay register 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ0BDLY3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEQ0BDLY3[15:0]**: PIE delay register 3

This bitfield is available for selection by the NOP and WAIT instructions in the PIE for the delay value. Programmed by dwc_ddrphy_phyinit_I_loadPIEImage() to support frequency changes.

Programming bit[3,2,1,0]:

- These bits are used to control various delays during PLL, DLL initialization.
- These bits must be programmed with the corresponding number of MEMCLK cycles representing a fixed amount of delay irrespective of DfiClk frequency.
- When calculating the number of cycles to be programmed, it must be noted that the sequencer multiplies programmed values by a factor of 8 and values are round down to the closest integer.

Bit: Required delay | Programmed value

- Bit[0]: 0.5 µs | (0.5 µs / MEMCLK period) / 8)
- Bit[1]: 1.0 µs | (1.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | (10.0 µs / MEMCLK period) / 8)
- Bit[2]: 10.0 µs | 64/4 if (MEMCLK freq ≤ 400 MHz)
- Bit[3]: - | 132/4 if (400 MHz freq < MEMCLK freq ≤ 533 MHz)
- Bit[3]: - | 176/4 if (MEMCLK freq > 533 MHz)

14.5.60 DDRPHYC PHY alert status bit register (DDRPHYC_MASTER0_PHYALERTSTATUS)

Address offset: 0x8003C

Reset value: 0xXXXX XXXX

This read-only register represents the status of the ALERT_N pin to the PHY.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYALERT
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PHYALERT**: Current state of ALERT_N

14.5.61 DDRPHYC setup intervals for DFI PHY master operation register (DDRPHYC_MASTER0_PPTTRAINSETUP)

Address offset: 0x80040

Reset value: 0x0000 0000

For LPDDR4 some tracking is required periodically using the DFI PHY MASTER interface. This register control the interval. Setting this register to 0 completely disables the MASTER. Programmed by dwc_ddrphy_phyinit_I_loadPIEImage() to support LPDDR4 DRAM drift compensation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYMSTRMAXREQTOACK[2:0]			PHYMSTRTRAININTERVAL[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **PHYMSTRMAXREQTOACK[2:0]**: Max time from tdfi_phymstr_req asserted to tdfi_phymstr_ack asserted.

If the request is not acknowledged by this time, dfi_error is asserted.
000: Disable PHY MASTER.
001: Set tPHYMSTR_resp. = 512 MEMCLKs.
010: Set tPHYMSTR_resp. = 1024 MEMCLKs.
011: Set tPHYMSTR_resp. = 2048 MEMCLKs.
100: Set tPHYMSTR_resp. = 4096 MEMCLKs.
101: Set tPHYMSTR_resp. = 8192 MEMCLKs.
110: Set tPHYMSTR_resp. = 32768 MEMCLKs.
111: Reserved

*Note: The above are nominal values. The actual values are reduced by (2 + (DWC_DDRPHY_PIPE_DFI_MISC * MEMCLK/DfiCtlClk)) * 2*

This signal interacts with dfi_phyupd_req.

Bits 3:0 **PHYMSTRTRAININTERVAL[3:0]**: Time between the end of one training, and the start of the next.

As per section 11 of the preliminary DFI 4.0 Specification,V2, it is the max expected time from dfi_init_complete asserted to tdfi_phymstr_ack asserted.
0000: Disable PHY MASTER.
0001: PHY MASTER request interval = 524288 MEMCLKs.
0010: PHY MASTER request interval = 1048576 MEMCLKs.
0011: PHY MASTER request interval = 2097152 MEMCLKs.
0100: PHY MASTER request interval = 4194304 MEMCLKs.
0101: PHY MASTER request interval = 8388608 MEMCLKs.
0110: PHY MASTER request interval = 16777216 MEMCLKs.
0111: PHY MASTER request interval = 33554432 MEMCLKs.
1000: PHY MASTER request interval = 67108864 MEMCLKs.
1001: PHY MASTER request interval = 134217728 MEMCLKs.
1010: PHY MASTER request interval = 268435456 MEMCLKs.
Other: Reserved

14.5.62 DDRPHYC ATestMode control register (DDRPHYC_MASTER0_ATESTMODE)

Address offset: 0x80048

Reset value: 0x0000 0000

Controls for address command loopback. This register is written by ATE firmware.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTMODESEL[2:0]			ATEST CLKEN	ATEST PRBSEN
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:2 **ATESTMODESEL[2:0]**: Master mode select for ATest (loopback)

000: Mission mode, all ATest disabled, loopback receivers powered down

001: External loopback mode [Single data rate pattern - dfi_cas sent to all lanes]

010: Internal loopback mode [Single data rate pattern]

011: Internal loopback mode [Double data rate pattern]

100: External loopback mode [Single data rate pattern - corresponding DFI signal sent to each lane]

Bit 1 **ATESTCLKEN**: Not used

Bit 0 **ATESTPRBSEN**:

Enables loopback PRBS7 testing of all the DDR output pins.

The asserting edge resets all the individual. TestPrbsErrCnt and there is no other reset for the TestPrbsErrCnt.

14.5.63 DDRPHYC TxCalBinP register (DDRPHYC_MASTER0_TXCALBINP)

Address offset: 0x80050

Reset value: 0xXXXX XXXX

TX P impedance calibration observation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINP[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **TXCALBINP[4:0]**: Binary result of the 31-bit thermometer pull-up code

14.5.64 DDRPHYC TxCalBinN register (DDRPHYC_MASTER0_TXCALBINN)

Address offset: 0x80054

Reset value: 0xXXXX XXXX

TX N impedance calibration observation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINN[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **TXCALBINN[4:0]**: Binary result of the 31-bit thermometer pull-down code

14.5.65 DDRPHYC TX P impedance calibration override register (DDRPHYC_MASTER0_TXCALPOVR)

Address offset: 0x80058

Reset value: 0x0000 0000

Override values for TxCalBinP register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINPOVREN	TXCALBINPOVRVAL[4:0]				
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TXCALBINPOVREN**:
1: Use the override value present in TxCalBinPOvrVal bit.
0: Reserved

Bits 4:0 **TXCALBINPOVRVAL[4:0]**: Binary value which can override TXCALBINP bitfiled Calibrator results if TXCALBINPOVREN bit is set.

14.5.66 DDRPHYC TX N impedance calibration override register (DDRPHYC_MASTER0_TXCALNOVR)

Address offset: 0x8005C

Reset value: 0x0000 0000

Override values for TXCALBINN bitfield.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINNOVREN	TXCALBINNOVRVAL[4:0]				
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TXCALBINNOVREN**:

1: Use the override value present in TXCALBINNOVRVAL bitfield.

0: reserved.

Bits 4:0 **TXCALBINNOVRVAL[4:0]**: Binary value which can override TXCALBINN bitfield Calibrator results if TXCALBINNOVREN bit is set.

14.5.67 DDRPHYC enables for update and low-power interfaces for DFI0 and DFI1 register (DDRPHYC_MASTER0_DFIMODE)

Address offset: 0x80060

Reset value: 0x0000 0001

Each bit enables the PHY update, PHY MASTER update, and low-power logic for the respective interfaces.

There are three valid configurations:

- 101: DFI0 is enabled and controls the entire PHY.
- 001: DFI0 controls 1/2 the PHY, external logic ties off inputs to DFI1.
- 011: DFI0 and DFI1 are active and are separately controlled.

This register interacts with the DWC_DDRPHY_DFI1_EXISTS define as follows:

- DfiMode = 001, DWC_DDRPHY_DFI1_EXISTS defined: ANIBs/DBYTEs assigned to DFI1 are idle.
- DfiMode = 011, DWC_DDRPHY_DFI1_EXISTS defined: DFI0 controls 1/2 PHY (CH. A), DFI1 controls 1/2 PHY (CH. B).
- DfiMode = 101, DWC_DDRPHY_DFI1_EXISTS defined: ANIBs/DBYTEs assigned to DFI1 are controlled by DFI0. Only DFI0 sideband signals are active.
- DfiMode = 001, DWC_DDRPHY_DFI1_EXISTS undefined: DFI0 controls entire PHY.
- DfiMode = 011, DWC_DDRPHY_DFI1_EXISTS undefined: illegal.
- DfiMode = 101, DWC_DDRPHY_DFI1_EXISTS undefined: not applicable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI1OVERRIDE	DFI1ENABLE	DFI0ENABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DFI1OVERRIDE**:
Controls the PHY logic associated with both DFI0 and DFI1.

Bit 1 **DFI1ENABLE**:
Enables operation for the PHY logic associated with DFI1.

Bit 0 **DFI0ENABLE**:
Enables operation for the PHY logic associated with DFI0.

14.5.68 DDRPHYC mode select register for MEMCLK/address/command tristate register (DDRPHYC_MASTER0_TRISTATEMODECA)

Address offset: 0x80064

Reset value: 0x0000 0000

This register is used to control the PHY dynamic address/command tristate feature, and the stopped MEMCLK state.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKDISVAL[1:0]		DDR2T MODE	DISDY NADRT RI
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **CKDISVAL[1:0]**:

The PHY provides four memory clocks (0 to 3).

- When the toggling of memory clock CK_t[n] is disabled with dfi_clk_disable[n] = 1, the memory clock CK_t[n] is driven with CkDisVal[1] bit.
- When the toggling of memory clock CK_c[n] is disabled with dfi_clk_disable[n] = 1, the memory clock CK_c[n] is driven with ~(CkDisVal[0] bit) (inverted).

Note: The non-toggling differential memory clock [CK_t[n],CK_c[n]] can be driven with any combination, LL,LH,HL,HH; the default [CK_t[n], CK_c[n]] = LH.

Bit 1 **DDR2TMODE**:

Must be set to 1 for dynamic tristate to work when CA bus is 2T or Geardown mode.
This bit has no effect when DISDYNADRTRI == 1.

Bit 0 **DISDYNADRTRI**:

When this bit is set, the dynamic tristate feature is disabled (on by default). This feature must be disabled for these modes:

- DR3/2T (if the controller cannot follow the 2T PHY tristate protocol)
- DDR4/2T/2N (if the controller cannot follow the 2T PHY tristate protocol)
- LPDDR4

When this bit =0 (default):

- In DDR4 mode, the following SDRAM pins can be dynamically tristated: A, BA, BG, ACT_n.
- In 1T mode, the PHY tristates the relevant pins when all ranks are deselected.
- In 2T mode (or geardown), the PHY tristates the relevant pins when:
  - DDR3: All ranks deselected (all CS_L==1) and [RAS_n,CAS_n,WE_n,BA0] = [1,1,1,0]
  - DDR4: All ranks deselected (all CS_L==1) and [RAS_n,CAS_n,WE_n,BA0,ACT_n] = [1,1,1,0,1]
    When in this mode, the controller must avoid sending the above patterns with any rank selected (for example: NOPs sent by the controller must have BA0 set to a non-zero value).

14.5.69 DDRPHYC MtestMuxSel register (DDRPHYC_MASTER0_MTESTMUXSEL)

Address offset: 0x80068

Reset value: 0x0000 0000

Digital observation pin control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MTESTMUXSEL[5:0]**:

Controls for the 64-1 mux for asynchronous data to the digital observation pin.
0x0: Drive 0 from this device (allows flat 'OR' of pass-through information).
0x01 to 0x1F: Select local data from AC/DBYTE/MASTER macro.
0x20: Drive 0 from PUB synth logic path into mux.
0x21 to 0x3F: Select local data from PUB AC/DBYTE/MASTER synthesized logic.

14.5.70 DDRPHYC MtestPgmInfo register (DDRPHYC_MASTER0_MTESTPGMINFO)

Address offset: 0x8006C

Reset value: 0x0000 0000

Digital observation pin program information for debug.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTPGMINFO
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MTESTPGMINFO**:

This bit value can be driven onto the digital observation pin (no other hardware effect).

14.5.71 DDRPHYC dynamic power up/down control register (DDRPHYC_MASTER0_DYNPWRDNUP)

Address offset: 0x80070

Reset value: 0x0000 0000

Register written by PIE and the data in here are overwritten.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DYNPOWERDOWN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DYNPOWERDOWN**:

1: Analog circuitry (voltage dacs, bias gen) is turned off.

0: Normal operation

14.5.72 DDRPHYC PMIEnable register (DDRPHYC_MASTER0_PMIENABLE)

Address offset: 0x80074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMIENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PMIENABLE**:

This bit is dynamically written by PIE during frequency changes and must not be written by the user.

14.5.73 DDRPHYC PhyTID register (DDRPHYC_MASTER0_PHYTID)

Address offset: 0x80078

Reset value: 0x0000 0000

PHY technology ID register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYTID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PHYTID[15:0]**: Placeholder to store technology-specific information

14.5.74 DDRPHYC HwtMRL register (DDRPHYC_MASTER0_HWTMRL)

Address offset: 0x80080

Reset value: 0x0000 0006

HWT MaxReadLatency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTMRL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTMRL[4:0]**: Master copy of MRL used by the PHY training firmware only

14.5.75 DDRPHYC DFI PhyUpdate request time counter register (DDRPHYC_MASTER0_DFIPHYUPD)

Address offset: 0x80084

Reset value: 0x0000 FF07

This register is used to determine the DFI update request interval (in MEMCLKs) and the maximum update response time. DFIPHYUPDCNT and DFIPHYUPDRESP must be chosen such that DFIPHYUPDCNT > DFIPHYUPDRESP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIPHYUPDINTTHRESHOLD[3:0]				DFIPHYUPDTHRESHOLD[3:0]				DFIPHYUPDMODE	DFIPHYUPDRESP[2:0]			DFIPHYUPDCNT[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIPHYUPDINTTHRESHOLD[3:0]**:

Similar to DFIPHYUPDTHRESHOLD except that, rather than affecting the Phy update request, this bitfield affects only the threshold used to generate the VT drift alarm interrupt. The actual threshold is twice what is specified in this bitfield.

Bits 11:8 **DFIPHYUPDTHRESHOLD[3:0]**:

0x0: Disable threshold-based Phy update requests when DFIPHYUPDMODE = 0x1.

- Nonzero codes are the threshold value for the change in the master LCDL 1UI phase code since the last Phy update request that triggers a new Phy update request;
- If (current_1UI_phase - last_1UI_phase) > DFIPHYUPDTHRESHOLD, then a Phy update is requested.

In units of native-control phase units, minimum value is 1.

With DFIPHYUPDTHRESHOLD = 1, a master LCDL dithering between two values does not cause a Phy update request.

Bit 7 **DFIPHYUPDMODE**:

0: Deterministic timer-based Phy update requests. Enables multi-channel/multi-phy lockstep operation (default, recommended).

1: Non-deterministic threshold-based Phy update requests. Multi-channel/multi-phy operation not permitted, all requests must be acknowledged.

Bits 6:4 **DFIPHYUPDRESP[2:0]**:

Enforces the t_phyupd_resp time (max time allowed to the controller to respond to the request for a PHY update). A dfi_error is signaled if there is no acknowledgement of the update request within nMEMCLKs_phyupd_resp.

000: nMEMCLKs_phyupd_resp= 1024 MEMCLKs (default)

001: nMEMCLKs_phyupd_resp= 2048 MEMCLKs

010: nMEMCLKs_phyupd_resp= 4096 MEMCLKs

011: nMEMCLKs_phyupd_resp= 8192 MEMCLKs (not allowed in LPDDR4 mode)

*Note: The above are nominal values. The actual values are reduced by (2 + (DWC_DDRPHY_PIPE_DFI_MISC * MEMCLK/DfiCtlClk)) * 2.*

Any code not enumerated above is reserved and must not be set.

This signal interacts with dfi_phymstr_req.

Bits 3:0 **DFIPHYUPDCNT[3:0]**:

Controls the interval between the end of a PHY update transaction and a subsequent request.

0000: Disable timer-based PHY update when DFIPHYUPDMODE= 0.

0001: 16K MEMCLKs minus nMEMCLKs_phyupd_resp

0011: 32K MEMCLKs minus nMEMCLKs_phyupd_resp

0111: 64K MEMCLKs minus nMEMCLKs_phyupd_resp

1111: 128K MEMCLKs minus nMEMCLKs_phyupd_resp

Note: Any code not enumerated above is reserved and must not be set.

14.5.76 DDRPHYC PdaMrsWriteMode register (DDRPHYC_MASTER0_PDAMRSWRITEMODE)

Address offset: 0x80088

Reset value: 0x0000 0000

Controls the write DQ generation for Per-Dram-Addressing of MRS.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDAMRSWRITEMODE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PDAMRSWRITEMODE**:

Controls the write DQ generation per the timing requirements on DQ signals used for Per-Dram-Addressing mode of MRS commands.

When asserted, the DQ are driven for an additional UI before the logical burst, and for an additional UI after the logical burst. The early UI has the value of first logical UI, and the late UI has the value of the last logical UI. The timing of the logical burst is unchanged.

No modification to TxDqDly bit is required or performed.

Note: It is strongly recommended to clear this bit after completing the MRS that clears MR3 A4 and for mission-mode writes, else required bubbles are larger and the power consumed is greater. See JEDEC MR3 A4, Per DRAM Addressability.

14.5.77 DDRPHYC DFIGEARDOWNCTL register (DDRPHYC_MASTER0_DFIGEARDOWNCTL)

Address offset: 0x8008C

Reset value: 0x0000 0000

Controls whether dfi_geardown_en causes CS and CKE timing to change.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIGEARDOWNCTL[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **DFIGEARDOWNCTL[1:0]**:

Bit[0] controls whether dfi_geardown_en causes chip-select (CS) timing to change.
0: CS timing is not dynamically modified with dfi_geardown_en.
1: CS is delayed by an additional 1UI dynamically when dfi_geardown_en==1.

Note: This bit must be set to zero if the PHY is not in DDR4 mode.

Bit[1] controls whether dfi_geardown_en causes clock-enable (CKE) timing to change.
0: CKE timing is not dynamically modified with dfi_geardown_en
1: CKE is delayed by an additional 1UI dynamically when dfi_geardown_en==1

Note: This bitfield must be set to zero if the PHY is not in DDR4 mode.

14.5.78 DDRPHYC control the PHY logic related to the read and write DQS preamble register (DDRPHYC_MASTER0_DQSPREAMBLECONTROL)

Address offset: 0x80090

Reset value: 0x0000 0008

For improved robustness in operating with DRAM timing drift, this register provides controls for the width of the write DQS preamble and postamble, and controls for the anticipated width of DQS preamble for reads.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDQSEXTENSION	LP4STCPREBRIDGERXEN	LP4POSTAMBLEEXT	LP4TGLTWOTCKTXDQSPRE	POSITIONDFEINIT[2:0]			TWOTCKTXDQSPRE	TWOTCKRXDQSPRE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **WDQSEXTENSION**:

When this bits is set, DQS_T and DQS_C are driven differentially to 0 and 1, respectively, before and after a write burst, except during a memory read transaction.

Note: Use of WDQSEXTENSION requires POdtTailWidth = 3 and POdtStartDelay = 00.

Bit 7 **LP4STTCPREBRIDGERXEN**: Used in LPDDR4 static-preamble mode to bridge the RxEn between two reads to the same timing group when the bubble is 1 memclk

LPDDR4 static-preamble mode is set in the DRAMs with MR1, OP[3] = 0.

Bit 6 **LP4POSTAMBLEEXT**:

In LPDDR4 mode, this bit must be set to extend the write postamble.

0: Half-memclk postamble, as in D4

1: One-and-one-half-memclk postamble

Note: See LPDDR4 Spec MR3, OP[1] WR PST, vendor-specific function.

Bit 5 **LP4TGLTWOTCKTXDQSPRE**: Used in LPDDR4 mode to modify the early preamble when TwoTckTxDqsPre = 1.

0: Level first-memclk preamble

1: Toggling first-memclk preamble

Bits 4:2 **POSITIONDFEINIT[2:0]**: For DDR4 PHY, only when receive DFE is enabled

0b010 is the nominal recommended value for leading edge used by receiver.

- Bit[2] = 1 causes DfeInit to be asserted late.
- Bit[1] = 1 causes DfeInit to be asserted at the nominal time.
- Bit[0] = 1 causes DfeInit to be asserted early.

Bit 1 **TWOTCKTXDQSPRE**:

For LPDDR4, all write operations are 2nCK such that this control must be set to 1.

Bit 0 **TWOTCKRXDQSPRE**:

Widens the RxDqsEn window to allow larger drift in the incoming read DQS to take advantage of the larger/wider preamble generated by the DRAMs when D4 DRAMS are configured with DDR4 MR4 A11 read preamble = 1 for causing a 2nCK read preamble.

For LPDDR4, all read operations are 2nCK such that this control must be set to 1.

Note: The nominal trained-to-center point is earlier relative to the first strobing edge of DQS than when not in this mode.

14.5.79 DDRPHYC MasterX4Config register (DDRPHYC_MASTER0_MASTERX4CONFIG)

Address offset: 0x80094

Reset value: 0x0000 0000

DBYTE module controls to select X4 DRAM device mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X4TG[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **X4TG[3:0]**:
Set to 1 if this timing group/rank is x4 (as opposed to x8) memory.
xxx[1|0]: TxReqDest/RxReqDest 0 X4/X8.
xx[1|0]x: TxReqDest/RxReqDest 1 X4/X8.
x[1|0]xx: TxReqDest/RxReqDest 2 X4/X8.
[1|0]xxx: TxReqDest/RxReqDest 3 X4/X8.
Note: This bit must be written before dfi_init_start is asserted.

14.5.80 DDRPHYC write level feedback DQ observability selection register (DDRPHYC_MASTER0_WRLEVBITS)

Address offset: 0x80098

Reset value: 0x0000 0098

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRLEVFORDQSU[3:0]				WRLEVFORDQSL[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **WRLEVFORDQSU[3:0]**: DQ bit used for write levelization
0000: use DQ0
..
0111: use DQ7
1000: use |DQ[3:0]
1001: use |DQ[7:4]
1010: use |DQ[7:0]

Bits 3:0 **WRLEVFORDQSL[3:0]**: DQ bit used for write levelization
0000: use DQ0
..
0111: use DQ7
1000: use |DQ[3:0]
1001: use |DQ[7:4]
1010: use |DQ[7:0]

14.5.81 DDRPHYC EnableCsMulticast register (DDRPHYC_MASTER0_ENABLECSMULTICAST)

Address offset: 0x8009C

Reset value: 0x0000 0000

In DDR4 mode, this controls whether CS_N[3:2] must be multicast on CID[1:0].

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLECSMULTICAST
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLECSMULTICAST**:

Controls whether CS_N[3:2] must be multicast on CID[1:0] in DDR4 mode.
0: Do not override pins corresponding to cid[1:0] (dfi_cid[1:0] connects to the pads).
1: Override pins corresponding to cid[1:0] with dfi_cs[3:2]. (cs[3:2] is multicast on cs[3:2] and cid[1:0]).

14.5.82 DDRPHYC Acx4AnibDis register (DDRPHYC_MASTER0_ACX4ANIBDIS)

Address offset: 0x800B0

Reset value: 0x0000 0000

Disable for unused ACX nibbles.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ACX4ANIBDIS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **ACX4ANIBDIS[11:0]**:

When a bit is set, the corresponding ACX nibble is disabled. Specifically, the I/O OE is disabled, as is the Dfi-side FIFO clock

14.5.83 DDRPHYC enable Read-DBI function in each DBYTE register (DDRPHYC_MASTER0_DMIPINPRESENT)

Address offset: 0x800B4

Reset value: 0x0000 0000

This register is used to instruct the PHY whether to enable the Read-DBI function on lane 8 of each DBYTE. Only the LPDDR4 and DDR4 protocols support the Read-DBI feature.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDDBIENABLED
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **RDDBIENABLED**:

This bit must be set if Read-DBI is enabled in a connected DDR4 or LPDDR4 device. If set, the following DRAM MR must also be set [DDR4.MR5.A12 = 1 or LPDDR4.MR3.OP[7] = 1].

14.5.84 DDRPHYC ARdPtrInitVal register (DDRPHYC_MASTER0_ARDPTRINITVAL)

Address offset: 0x800B8

Reset value: 0x0000 0000

Address/command FIFO read pointer initial value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARDPTRINITVAL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **ARDPTRINITVAL[3:0]**: Initial pointer offset for the free-running FIFOs in the DBYTE and ACX4 hardips

The units of this bitfield are in UI. The pointer separation must be chosen to compensate for all sources of skew and drift of the PHY DFICLK and PCLK networks.

This bitfield must be programmed in Step C of the PHY Initialization sequence.

14.5.85 DDRPHYC DLL mode control for DBYTEs register (DDRPHYC_MASTER0_DBYTEDLLMODECNTRL)

Address offset: 0x800E8

Reset value: 0x0000 0000

DLL mode control register in DBYTE modules.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLLRX PREA MBLE MODE	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DLLRXPREAMBLEMODE**:
This bit must be set to 1 if read DQS preamble contains a toggle, for example DDR4 or LPDDR4 read toggling preamble mode.
In LPDDR4, the PIE always sets this bit field to 0x1 for correct operation of the retraining algorithm.

Bit 0 Reserved, must be kept at reset value.

14.5.86 IDDRPHYC impedance calibration offsets control register (DDRPHYC_MASTER0_CALOFFSETS)

Address offset: 0x80114

Reset value: 0x0000 0000

Safety register that can be used to correct systematic offsets from each calibration stage. It is not recommended to change these values from the default zero value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CALDRVPUTHOFFSET[3:0]				CALDRVPDTHOFFSET[3:0]				CALCMPR5OFFSET[5:0]					
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:10 **CALDRVPUTHOFFSET[3:0]**:
Adjusts the driver pull-up calibration code.

Bits 9:6 **CALDRVPDTHOFFSET[3:0]**:
Adjusts the driver pull-down calibration code.

Bits 5:0 **CALCMPR5OFFSET[5:0]**:
Adjusts the offset-compensated DAC code for the cmpana circuit at VREF == 0.5 * VDDQ.

14.5.87 DDRPHYC SarInitVals register (DDRPHYC_MASTER0_SARINITVALS)

Address offset: 0x8011C

Reset value: 0x0000 0127

This register specifies the initial value for the compensation engine. It is not recommended to change the default value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SARINITPEXT[2:0]			SARINITNINT[2:0]			SARINITOFFSET05[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **SARINITPEXT[2:0]**:
Specifies the SAR starting value for PEXT calibration. This is a 5-bit wide target.

Bits 5:3 **SARINITNINT[2:0]**:
Specifies the SAR starting value for NINT calibration. This is a 5-bit wide target.

Bits 2:0 **SARINITOFFSET05[2:0]**:
Specifies the SAR starting value for OFFSET05 calibration. This is an 8-bit wide target.

14.5.88 DDRPHYC CalPExtOvr register (DDRPHYC_MASTER0_CALPEXTOVR)

Address offset: 0x80124

Reset value: 0x0000 0000

Impedance calibration PExt override control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALPEXTOVR[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CALPEXTOVR[4:0]**:
If CALPEXTDIS is set, the value provided here by software is used instead of the automatically generated value which is visible via CalPExt field.
This bitfield can only be written when the calibrator is not running.

14.5.89 DDRPHYC CalCmpr5Ovr register (DDRPHYC_MASTER0_CALCMPR5OVR)

Address offset: 0x80128

Reset value: 0x0000 0000

Impedance calibration Cmpr 50 control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALCMPR5OVR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CALCMPR5OVR[7:0]**:

If CALCMPR5DIS is set, the value provided here by software is used instead of the automatically generated value which is visible via CalCmpr5 field.
This bitfield can only be written when the calibrator is not running.

14.5.90 DDRPHYC CalNIntOvr register (DDRPHYC_MASTER0_CALNINTOVR)

Address offset: 0x8012C

Reset value: 0x0000 0000

Impedance calibration NInt override control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALNINTOVR[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CALNINTOVR[4:0]**:

If the CALNINTDIS is set, the value provided here by software is used instead of the automatically generated value which is visible via CalNInt field.
This bitfield can only be written when the calibrator is not running.

14.5.91 DDRPHYC impedance calibration driver strength control register (DDRPHYC_MASTER0_CALDRVSTR0)

Address offset: 0x80140

Reset value: 0x0000 0000

For a given calibration target, this register specifies the drive strength(s) to be used. There is no encoding. The strength selected must match the external resistor value used for calibration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALDRVSTRPU50[3:0]				CALDRVSTRPD50[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CALDRVSTRPU50[3:0]**:
0: 240.0R (Default)
1: 120.0R
2: 40.0R
3 to 15: Reserved

Bits 3:0 **CALDRVSTRPD50[3:0]**:
0: 240.0R (Default)
1: 120.0R
2: 40.0R
3 to 15: Reserved

14.5.92 DDRPHYC read data on-die termination timing control register (DDRPHYC_MASTER0_PROCODTTIMECTL)

Address offset: 0x80158

Reset value: 0x0000 001F

Controls timing of ProcOdt for DQ and DQS termination during reads (by PHY). Enabling and disabling of ProcOdt can be shifted with a granularity of 1UI. If WDQSEXTENSION is enabled, this register must be set to 0x03. Otherwise, use the recommended settings based

on MEMCLK frequency and TwoTckRxDqsPre. This ensures that ProcOdt is enabled before the DQS preamble begins, and is not disabled until after the DQS postamble ends.

Recommended configuration if TwoTckRxDqsPre == 1:

- 0 MHz < memclk-freq ≤ 933, PODTSTARTDELAY[1:0] = 2, PODTTAILWIDTH[1:0] = 2
- 933 MHz < memclk-freq ≤ 1200, PODTSTARTDELAY[1:0] = 0, PODTTAILWIDTH[1:0] = 2
- 1200 MHz < memclk-freq ≤ NA, PODTSTARTDELAY[1:0] = 0, PODTTAILWIDTH[1:0] = 3

Recommended configuration if TwoTckRxDqsPre == 0:

- 0 MHz < memclk-freq ≤ 933, PODTSTARTDELAY[1:0] = 2, PODTTAILWIDTH[1:0] = 2
- 933 MHz < memclk-freq ≤ 1200, PODTSTARTDELAY[1:0] = 1, PODTTAILWIDTH[1:0] = 2
- 1200 MHz < memclk-freq < PODTSTARTDELAY[1:0] = 3

See related DDRPHYC_DBYTEy_TXODTDRVSTREN_By registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PODTTAILWIDTHEXT[1:0]		PODTSTARTDELAY[1:0]		PODTTAILWIDTH[1:0]	
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **PODTTAILWIDTHEXT[1:0]**:

Controls the length of the tail of ProcOdt (units of UI). Controls the time from closing the window to receive DQS to ProcOdt deassertion. This bitfield can be set with values ranging from 0 to 3.

The aforementioned timing is defined as
(3 + PODTTAILWIDTH + PODTTAILWIDTHEXT - TwoTckRxDqsPre) UIs.

- It is recommended to use this bitfield as an extension to POdtTailWidth, and has a default value of 1. Note that this might not be the optimal value, since it is technology dependent.

Bits 3:2 **PODTSTARTDELAY[1:0]**:

Controls the start of ProcOdt (units of UI).

11: delay start 2 UI (maximum delay of start of ProcOdt)

10; delay start 1 UI

01: delay start 0 UI (default)

00: early by 1 UI

The time from ProcOdt assertion to open the window to receive DQS is than
(10 - PODTSTARTDELAY) UI. It can be up to 4 pclk smaller if small coarse delays are used (only at the lowest clock frequencies).

Bits 1:0 **PODTTAILWIDTH[1:0]**:

Controls the length of the tail of ProcOdt (units of UI). Controls the time from closing the window to receive DQS to ProcOdt deassertion. This bitfield can be set with values ranging from 0 to 3.

The aforementioned timing is defined as
(3 + PODTTAILWIDTH + PODTTAILWIDTHEXT - TwoTckRxDqsPre) UIs.

It is recommended to use this bitfield as the main timing configuration unit, with default value of 3. This may not be the optimal value, since it is technology dependent.

14.5.93 DDRPHYC MemAlert receiver configuration register (DDRPHYC_MASTER0_MEMALERTCONTROL)

Address offset: 0x8016C

Reset value: 0x0000 4000

Controls for the receiver for MemAlert (D4 parity or WCRC error) or, alternately, for ERR_n (D3 registered DIMM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MALERTFORCEERROR	MALERTDISABLEVAL	MALERTRXEN	MALERTPUEN	MALERTPUSTREN[3:0]				MALERTVREFEXTEN	MALERTVREFLEVEL[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MALERTFORCEERROR**:

When set, this bit state is used to force the parity error to memory.

Bit 14 **MALERTDISABLEVAL**:

When MALERTRXEN is not set, this bit state is used to drive dfi_alert_n.

Bit 13 **MALERTRXEN**:

This bit is programmable as follows:
0: Disable the receiver and send MALERTDISABLEVAL to dfi_alert_n.
1: Enable the receiver, and received data are forwarded to dfi_alert_n.

Bit 12 **MALERTPUEN**:

When set, this bit enables the pull-up termination on MALERT

Bits 11:8 **MALERTPUSTREN[3:0]**:
Controls the pull-up termination on MALERT.
- bit[8]: Controls a 240 Ω pull-up leg.
- bit[9]: Controls a 240 Ω pull-up leg.
- bit[10]: Controls a 120 Ω pull-up leg.
- bit[11]: Controls a 120 Ω pull-up leg.

This bitfield is programmable as follows:
0000: No pull-up strength
0001: 240 Ω pull-up strength
0010: 240 Ω pull-up strength
0011 - 120 Ω pull-up strength
0100: 120 Ω pull-up strength
0101: 80 Ω pull-up strength
0110: 80 Ω pull-up strength
0111: 60 Ω pull-up strength
1000: 120 Ω pull-up strength
1001: 80 Ω pull-up strength
1010: 80 Ω pull-up strength
1011: 60 Ω pull-up strength
1100: 60 Ω pull-up strength
1101: 48 Ω pull-up strength
1110: 48 Ω pull-up strength
1111: 40 Ω pull-up strength

Bit 7 **MALERTVREFEXTEN**:
When set for test/debug, this bit selects external V_{REF} source. It must not be set in mission mode.

Bits 6:0 **MALERTVREFLEVEL[6:0]**:
Sets the V_{REF} level of internal V_{REF} DAC, assuming the following mission mode settings: [TestExtVrefRange,TestMajorMode] = 0011 or 0000.
The V_{REF} level chosen by the receiver follows the equation:
$V_{REF} = V_{DDQ} * (0.51 + MALERTVREFLEVEL[6:0] * 0.00345)$.
The min/max values possible are:
- V_{REF} min = $V_{DDQ} * 0.51$
- V_{REF} max = $V_{DDQ} * 0.948$

14.5.94 DDRPHYC MemAlert receiver configuration register 2 (DDRPHYC_MASTER0_MEMALERTCONTROL2)

Address offset: 0x80170

Reset value: 0x0000 0000

Controls for the MemAlert output (dfi_alert_n) The DFI interface defines that the signal dfi_alert_n as a synchronous output on DFICLK. For some memory systems, this signal is

synchronized again elsewhere in the device. The synchronizer in the PHY is then useless (and simply added to the latency of observing an ALERT transition).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MALERTSYNCBYPASS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MALERTSYNCBYPASS**:

This bit is programmed as follows:

0: The phy drives dfi_alert_n with a synchronized value of the ALERT_N receiver.

1: The phy drives dfi_alert_n with the asynchronous value of the ALERT_N receiver.

14.5.95 DDRPHYC protection and control of BP_MemReset_L register (DDRPHYC_MASTER0_MEMRESETL)

Address offset: 0x80180

Reset value: 0x0000 0000

Controls the MemResetL output of the PHY. Intended to be used to protect/force the BP_MemReset_L output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTECTMEMRESET	MEMRESETLVALUE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PROTECTMEMRESET**:

Controls the MemResetL output of the PHY. The state of the MemResetL is transfered to the PHY BP_MemReset_L output when this bit is set.

Bit 0 **MEMRESETLVALUE**:

Controls the MemResetL output of the PHY. The state of this bit is transfered to the PHY BP_MemReset_L output when PROTECTMEMRESET is set.

14.5.96 DDRPHYC PUBMODE - HWT Mux select register (DDRPHYC_MASTER0_PUBMODE)

Address offset: 0x801B8

Reset value: 0x0000 0000

Selects between HWT and DCT as source of memory transactions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTMEMSRC
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HWTMEMSRC**:

When this bit is set, the mux that switches between DCT and HWT for the source of memory transactions is switched to HWT.

0: Control to the DCT

1: Control to the HWT

The transition can only be made when the FIFOs are all in PtrInit (all PtrInit signals are high).

14.5.97 DDRPHYC misc PHY status bits register (DDRPHYC_MASTER0_MISCPHYSTATUS)

Address offset: 0x801BC

Reset value: 0xXXXX XXXX

This read-only register represents the miscellaneous status of the PHY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORMEMRESET	DCTSANE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PORMEMRESET**:

Returns the active-high value used by the custom circuit which drives the memory reset signal. To comply with JEDEC specifications for memory reset, this signal powers on 1. This signal is reset during the PHY initialization sequence.

Bit 0 **DCTSANE**:

Returns the status of the custom circuit which protects the MemResetL output of the PHY on the initial power-on or reset.

0: Hardware controls the MemResetL output.

1: The memory reset is being controlled by MEMRESESETLVALUE and PROTECTMEMRESET or dfi_reset_n.

14.5.98 DDRPHYC CoreLoopbackSel register (DDRPHYC_MASTER0_CORELOOPBACKSEL)

Address offset: 0x801C0

Reset value: 0x0000 0000

Controls whether the loopback path bypasses the final PAD node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CORELOOPBACKSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CORELOOPBACKSEL**:

This bit is controlled by the PHY test firmware. It enables core-side loopback operation of the PHY.

When set, the PHY PAD node is bypassed, and pre-pad node is selected for sampling by the PHY RX. Setting this bit enables full-speed loopback operation of the PHY regardless of the loading of the pad.

This bit must be set to 0x0 for mission mode operation (or pad-side loopback).

14.5.99 DDRPHYC DLL various training parameters register (DDRPHYC_MASTER0_DLLTRAINPARAM)

Address offset: 0x801C4

Reset value: 0x0000 0002

DLL training parameters

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTENDPHDTIME[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EXTENDPHDTIME[1:0]**: Used by the PHY firmware locking the LCDL delay cells
Number of extra DfiClk periods (= 4 * UI) for the DLL phase-detector outputs to become not-metastable

14.5.100 DDRPHYC HwtLpCsEnBypass register (DDRPHYC_MASTER0_HWTLPCSENBYPASS)

Address offset: 0x801D0

Reset value: 0x0000 0000

CSn disable bypass for LPDDR4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTLPCSENBYPASS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HWTLPCSENBYPASS**:
When set, this bit disables LpCsEn function for LPDDR4.

14.5.101 DDRPHYC DFI command/address mode register (DDRPHYC_MASTER0_DFICAMODE)

Address offset: 0x801D4

Reset value: 0x0000 0000

This register controls the PHY signal-to-bump-mapping mode. The default state is all bits 0. Setting bits changes the arch mode, and are required to be 1-hot. While all bits are present in all products, some are not always relevant as a particular DRAM type may not be supported. Irrelevant bits must always be set to 0. Programmed by dwc_ddrphy_phyinit_C_initPhyConfig() according to PHY configuration and DRAM type.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFID4ALTCAMODE	DFILP4CAMODE	DFID4CAMODE	DFILP3CAMODE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **DFID4ALTCAMODE**: DDR4-Alt mode enable (only available in certain configurations)
0: D4-Alt mode disabled
1: D4-Alt mode enabled

Bit 2 **DFILP4CAMODE**: LP4 mode enable
0: LP4 mode disabled
1: LP4 mode enabled

Bit 1 **DFID4CAMODE**: DDR4 mode enable
0: D4 mode disabled
1: D4 mode enabled

Bit 0 **DFILP3CAMODE**: Reserved, must be kept at reset value.

14.5.102 DDRPHYC DLL lock state machine control register (DDRPHYC_MASTER0_DLLCONTROL)

Address offset: 0x801E0

Reset value: 0x0000 0003

This register is written by the PIE, and data in here are overwritten.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLLRESETRSVD	DLLRESETSLAVE	DLLRESETRELOCK
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DLLRESETRSVD**: Reserved

Bit 1 **DLLRESETSLAVE**: Reserved.

Bit 0 **DLLRESETRELOCK**: Used to reset the DDL/LCDL lock state machine
Deasserting starts locking sequence. This must be done anytime the DLLs are to be relocked (for example: frequency change).

14.5.103 DDRPHYC DLL update phase control register (DDRPHYC_MASTER0_PULSEDLLUPDATEPHASE)

Address offset: 0x801E4

Reset value: 0x0000 0000

This register is used only for test and for training. It is not used in PIE frequency change or power-state change. The CfgWE for this register causes an UpdatePhase (ClkPhase) for the DLL associated with the subfield enables.

- Writing 1 to one of the PULSExxxDLLUPDATEPHASE causes a pulse. It asserts ClkPhase = 1, then clears ClkPhase = 0.
- There is no need to write a 0 after writing a 1 to one of the PulseXDllUpdatePhase.
- A written 1 reads back a 1, but that does not mean that ClkPhase to the corresponding DLL is being held high.
- Writing this register must be in a traffic bubble.
- Writing all-0s to this register is a NOP for the DLLs. It clears any read-back values if they are set.
- This function requires AlwaysEnCfgClk = 1.

Example to update the active phases of the DBYTE module DLLs (NOP for AC module DLLs):

```
io_write16(c0 | tMASTER | csr_CalAndPllConfig_ADDR,
csr_AlwaysEnCfgClk_MASK);
io_write16(cbrd | tANIB | csr_PulseDllUpdatePhase_ADDR,
csr_PulseDbyteDllUpdatePhase_MASK);
io_write16(cbrd | tANIB | csr_PulseDllUpdatePhase_ADDR, 0x0);
```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALWAYSUPDATELCDLPHASE	TRAINUPDATEPHASEONLONGBUBBLE	UPDATEPHASEDESTRESERVED[2:0]			PULSEACADLLUPDATEPHASE	PULSEACKDLLUPDATEPHASE	PULSEDBYTEDLLUPDATEPHASE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ALWAYSUPDATELCDLPHASE**:
Causes each new operation to reload the LcdlPhase, and increases bubbles.

Bit 6 **TRAINUPDATEPHASEONLONGBUBBLE**:
Causes LongBubble to update the DBYTE and ANIB LDCL phase. Can be used for training, but not to be used for mission mode.

Bits 5:3 **UPDATEPHASEDESTRESERVED[2:0]**: reserved, not used

Bit 2 **PULSEACADLLUPDATEPHASE**:
Causes an AC module CA (command/address/cke/odt) DLL phase update.

Bit 1 **PULSEACKDLLUPDATEPHASE**:

Causes an AC module CK (memck) DLL phase update. This bit must be set only when the memory CKE = 0 or memory RST = 0, or when there is no memory, as in tester mode.

Bit 0 **PULSEDBYTEDLLUPDATEPHASE**:

Causes a LongBubble to the DBYTE modules, which causes an update of the DBYTE module DLLs (tx,rxen,rxclk).

14.5.104 DDRPHYC DLL gain control register (DDRPHYC_MASTER0_DLLGAINCTL)

Address offset: 0x801F0

Reset value: 0x0000 00A1

Controls for the DllGain sequencing for locking and maintaining lock of the LCDL delay cells.

- The DllGain sequencer starts locking with the DllGainIV.
- As the end of locking approaches, the DllGain is incremented in stages until, at the end of locking, the DllGain is the terminal values DllGainTV.
- 2^(5-DllGain) is fractional change in the DAC code MstPhase1UI[8:0] of DLL per pclk.
- Firmware uses values for DllGainIV, DllGainTV, and LcdlSeed0 that are optimized for minimum locking time. These controls are not to be configured by the user.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DLLSEEDSEL[3:0]				DLLGAINTV[3:0]				DLLGAINIV[3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **DLLSEEDSEL[3:0]**: Reserved, must be configured to be 0.

Bits 7:4 **DLLGAINTV[3:0]**: Terminal value of DllGain (value in effect when locking is done), and value used for maintaining lock (tracking pclk variation)

DllGainTV must be greater than or equal to DllGainIV. The maximum value is 10.

Bits 3:0 **DLLGAINIV[3:0]**: Initial value of DllGain

14.5.105 DDRPHYC impedance calibration control register (DDRPHYC_MASTER0_CALRATE)

Address offset: 0x80220

Reset value: 0x0000 0009

Controls the impedance calibration engine in the PHY.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLEBACKGROUNDZQUPDATES	CALONCE	CALRUN	CALINTERVAL[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DISABLEBACKGROUNDZQUPDATES**: Reserved for debug only

When this bit is set, ZQ compensation code updates occur only during a DFI sideband [PHYUPDATE,CTRLUPDATE,PHYMSTR] handshake.

This debug feature must be disabled if DFI1 is active.

Bit 5 **CALONCE**:

The setting of this bit changes the behavior of CALRUN.

0: The calibration proceeds at the rate determined by CALINTERVAL[3:0].

1: The 0->1 transition of CALRUN causes a single iteration of the calibration sequence to occur. Once the calibration is complete, the calibration engine remains idle until another CALRUN trigger.

This bit must only be changed while the calibrator is idle (before CALRUN is set).

Bit 4 **CALRUN**:

This bit is programmed as follows:

0: The calibrator does not run. If already active, then it completes the current sequence before quiescing (default value because the calibrator needs to know the correct operating mode before it starts up).

1: A calibration sequence is triggered by the 0->1 transition of this bit, as determined by CALONCE.

This bit does not make anything while CALZAP = 1.

Bits 3:0 **CALINTERVAL[3:0]**: Interval between successive calibrations
0000: 0 ms (continuous)
0001: 0.010 ms
0010: 0.10 ms
0011: 1 ms
0100: 2 ms
0101 3 ms
0110: 4 ms
0111: 8 ms
1000: 10 ms
1001: 20 ms
Others: Reserved
This bitfield must only be changed while the calibrator is idle (before CALRUN is set).
The calibration intervals are valid only at DFICLK frequency ≥ 24 MHz.
If the DFICLK frequency is less than 24 MHz, the interval cab be up to 2.4x larger than the value programmed in this bitfield.

14.5.106 DDRPHYC CalZap register (DDRPHYC_MASTER0_CALZAP)

Address offset: 0x80224

Reset value: 0x0000 0000

Impedance calibration zap/reset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALZAP
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CALZAP**:
This bit is written by the PIE, and the data in here are overwritten. Setting this bit resets the calibrator to its idle state. This bit must be cleared for the calibration engine to run again.

14.5.107 DDRPHYC PState register (DDRPHYC_MASTER0_PSTATE)

Address offset: 0x8022C

Reset value: 0x0000 000C

PSTATE selection

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTATE[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PSTATE[3:0]**:

This bitfield is written by the PIE, and the data in here are overwritten.
The current PState inferred by the PIE from handshake requests on DFI status interface.
0000: PState P0
1100: LP2
1111: LP3
Others: Reserved
The lower two bits of this bitfield are used to select the PHY current PState for all PStateable registers.

Note: The value selected in either LP2 or LP3 is don't care because no transactions are taking place when in these states.

14.5.108 DDRPHYC PLL output control register (DDRPHYC_MASTER0_PLLOUTGATECONTROL)

Address offset: 0x80234

Reset value: 0x0000 0000

This register is written by the PIE, and the data in here are overwritten.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESERVED2X1	PCLKGATEEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RESERVED2X1**: Reserved

Bit 0 **PCLKGATEEN**: Reserved.

14.5.109 DDRPHYC PMU power-on reset control register (DDRPHYC_MASTER0_PORCONTROL)

Address offset: 0x80240

Reset value: 0x0000 0000

POR control (PLL/DLL lock done). This register is written by the PIE during PHY operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLDLLLOCKDONE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLDLLLOCKDONE**:

This bit is set by the PIE during execution, and cleared on PHY reset, or on power cycle. When set, this bit releases control of PHY I/Os from power-on defaults.

Note: This is a legacy bit name and no longer represents when PHY internal PLL or DLL is locked.

14.5.110 DDRPHYC CalBusy register (DDRPHYC_MASTER0_CALBUSY)

Address offset: 0x8025C

Reset value: 0xXXXX XXXX

Impedance calibration busy status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALBUSY
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CALBUSY**:

Read 1 if the calibrator is actively calibrating. Any changes to calibrator-related bits can only be made if the calibrator is disabled (via CALRUN), and this bit is read as 0.

14.5.111 DDRPHYC misc impedance calibration control register (DDRPHYC_MASTER0_CALMISC2)

Address offset: 0x80260

Reset value: 0x0000 0004

Second set of miscellaneous impedance calibrator controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED15	CALSLOWCMPANA	CALCANCELROUNDERRDIS	CALCMPTRRESTRIM	RESERVED11	RESERVED10X3[7:0]								CALNUMVOTES[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RESERVED15**: Miscellaneous impedance calibration controls

Bit 14 **CALSLOWCMPANA**:

When set, this bit doubles the time allowed for the analog comparator cell to settle before sampling begins.

Bit 13 **CALCANCELROUNDERRDIS**:

The PEXT and NINT calibration results naturally include a rounding error which manifests as a change of impedance at the pad. Where these errors are in the same direction, the PEXT calibration result is adjusted by 1 to offset NINT error. Setting this bit to 1 prevents this adjustment.

Bit 12 **CALCMPTRRESTRIM**: Reserved

Bit 11 **RESERVED11**: Reserved (formerly used to skip ODT calibration if required)

Bits 10:3 **RESERVED10X3[7:0]**: Reserved

Bits 2:0 **CALNUMVOTES[2:0]**:

This bitfield controls the number of consecutive comparator output bits over which majority voting is done. It effectively works as a sampling window, since each sample serves as a single vote. The result of the voting is fed into the successive approximation register (SAR). The voting is initiated after the time configured via CALSLOWCMPANA has elapsed. The number of votes is given below.

000: 1
001: 9
010: 17
011: 33
100: 65 (default)
101: 129
110: 193
111: 255

14.5.112 DDRPHYC controls for disabling the impedance calibration of certain target register (DDRPHYC_MASTER0_CALMISC)

Address offset: 0x80268

Reset value: 0x0000 0000

Table of impedance calibration bit relatedness:

Target: OFFSET05

- also known as: CALCMPR5
- working state: RESCALCMPR5
- r/w viewing bit: CALCMPR5
- override bit: CALCMPR5OVR
- offset bit: CALCMPR5OFFSET

Target: NINT

- also known as: CALDRVPD5
- working state: RESCALNINT
- r/w viewing bit: CALNINT
- override bit: CALNINTOVR
- offset bit: [CALDRVPDOFFSET, CALDRVPDTHOFFSET]

Target: PEXT

- also known as: CALDRVPU5
- working state: RESCALPEXT
- r/w viewing bit: CALPEXT
- Override bit: CALPEXTOVR
- Offset bit: [CALDRVPUOFFSET, CALDRVPUTHOFFSET]

Miscellaneous calibration control knobs. These values can only be changed when the calibrator is not running.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALPEXTDIS	CALNINTDIS	CALCMPR5DIS
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CALPEXTDIS**:

Setting this bit prevents the calibration engine from overwriting TXCALBINP and TXCALTHP with an automatically generated value, in which case a value must be supplied by software.

Bit 1 **CALNINTDIS**:

Setting this bit prevents the calibration engine from overwriting TXCALBIN and TXCALTHN with an automatically generated value, in which case a value must be supplied by software.

Bit 0 **CALCMPR5DIS**:
Setting this bit prevents the calibration engine from using the result from the CALCMPR5 stage of calibration. Software must supply a replacement value via CALCMPR5OVR.

14.5.113 DDRPHYC CalVRef register (DDRPHYC_MASTER0_CALVREFS)

Address offset: 0x8026C

Reset value: 0x0000 0002

VREF code for comparator offset compensation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALVREFS[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **CALVREFS[1:0]**:
This bitfield drives the Cmpdig_CalRef pin of the cmpana cell at various stages of calibration. Following are the encoding values
10: 50% V_{REF} (default)
Others: Reserved

14.5.114 DDRPHYC CalCmpr5 register (DDRPHYC_MASTER0_CALCMPR5)

Address offset: 0x80270

Reset value: 0xXXXX XXXX

Impedance calibration Cmpr control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALCMPR5[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CALCMPR5[7:0]**: Offset-compensated DAC code for the cmpana circuit at V_{REF} = 0.5 * V_{DDQ}.

14.5.115 DDRPHYC impedance calibration NInt control register (DDRPHYC_MASTER0_CALNINT)

Address offset: 0x80274

Reset value: 0xXXXX XXXX

Observation of calibration engine pulldown calibration code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALNINTTHB[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CALNINTTHB[4:0]**: Number of thermometer bits which are set

14.5.116 DDRPHYC impedance calibration PExt control register (DDRPHYC_MASTER0_CALPEXT)

Address offset: 0x80278

Reset value: 0xXXXX XXXX

Observation of calibration engine pull-up calibration code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALPEXTTHB[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CALPEXTTHB[4:0]**: Number of thermometer bits which are set

14.5.117 DDRPHYC impedance calibration Cmp invert control register (DDRPHYC_MASTER0_CALCMPINVERT)

Address offset: 0x802A0

Reset value: 0x0000 0003

A safety bit for each calibration target, which controls the inversion (or not) of the comparator output before it is sampled. Writing 1 in one bit causes the comparator output to be inverted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMPINVERTCALODTPU	CMPINVERTCALODTPD	CMPINVERTCALDRVPU50	CMPINVERTCALDRVPD50	CMPINVERTCALDAC50
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMPINVERTCALODTPU**: Impedance calibration Cmp invert control

Bit 3 **CMPINVERTCALODTPD**: Impedance calibration Cmp invert control

Bit 2 **CMPINVERTCALDRVPU50**: Impedance calibration Cmp invert control

Bit 1 **CMPINVERTCALDRVPD50**: Impedance calibration Cmp invert control

Bit 0 **CMPINVERTCALDAC50**: calibration Cmp invert control

14.5.118 DDRPHYC impedance calibration Cmpana control register (DDRPHYC_MASTER0_CALCMPANACNTRL)

Address offset: 0x802B8

Reset value: 0x0000 0112

This register controls the cmpana custom circuit analog. It is not recommended to change its values from the reset values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CMPRBIASBYPASSEN	CMPRGAINRESADJ	CMPRGAINCURRADJ[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CMPRBIASBYPASSEN**: Impedance calibration Cmpana control

Bit 8 **CMPRGAINRESADJ**: Impedance calibration Cmpana control

Bits 7:0 **CMPRGAINCURRADJ[7:0]**: Impedance calibration Cmpana control

14.5.119 DDRPHYC maps dfi_rddata_cs_n to destination dimm timing group register (DDRPHYC_MASTER0_DFIRDDATACSDESTMAP)

Address offset: 0x802C0

Reset value: 0x0000 00E4

This register must be default values in LPDDR4 mode. Maps dfi_rddata_cs_n (read chip-select asserted low) to destination dimm timing group.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIRDDESTM3[1:0]		DFIRDDESTM2[1:0]		DFIRDDESTM1[1:0]		DFIRDDESTM0[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **DFIRDDESTM3[1:0]**:
Maps dfi_rddata_cs_n[3] to destination DfiRdDestm3 timing.
For example, if 3 dfi_rddata_cs_n[3] uses Register Rx[En,Clk]DlyTg3 timing.

Bits 5:4 **DFIRDDESTM2[1:0]**:
Maps dfi_rddata_cs_n[2] to destination DfiRdDestm2 timing.
For example, if 2 dfi_rddata_cs_n[2] uses Register Rx[En,Clk]DlyTg2 timing.

Bits 3:2 **DFIRDDESTM1[1:0]**:
Maps dfi_rddata_cs_n[1] to destination DfiRdDestm1 timing.
For example, if 1 dfi_rddata_cs_n[1] uses Register Rx[En,Clk]DlyTg1 timing.

Bits 1:0 **DFIRDDESTM0[1:0]**:
Maps dfi_rddata_cs_n[0] to destination DfiRdDestm0 timing.
For example, if 0 dfi_rddata_cs_n[0] uses Register Rx[En,Clk]DlyTg0 timing.

14.5.120 PHY global V_{REF} controls register (DDRPHYC_MASTER0_VREFINGLOBAL)

Address offset: 0x802C8

Reset value: 0x0000 0200

This register controls the PHY global V_{REF} generator, and the PHY analog pad BP_VREF. The mission mode function of the PHY V_{REF} generator is to generate V_{REF}DQ for the PHY

RX function. This register is written and configured by the PHY training firmware for mission mode function.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GLOBALVREFINMODE	GLOBALVREFINTRIM[3:0]				GLOBALVREFINDAC[6:0]							GLOBALVREFINSEL[2:0]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **GLOBALVREFINMODE**: Reserved

Bits 13:10 **GLOBALVREFINTRIM[3:0]**: Reserved

Bits 9:3 **GLOBALVREFINDAC[6:0]**: DAC code for internal V_{REF} generation
The DAC has two ranges; the range is set by GLOBALVREFINSEL[2]
Range0: [GLOBALVREFINSEL[2] = 0] (TBD)
– DAC output voltage = GLOBALVREFINDAC == 0x00 then set to Hi-Z, else set to 0.345 * V_{DDQ} + (0.005 * GLOBALVREFINDAC * V_{DDQ})
Range1: [GLOBALVREFINSEL[2] = 1]
– DAC output voltage = GLOBALVREFINDAC == 0x00 then set to Hi-Z, else set to 0.005 * (GLOBALVREFINDAC - 1) * V_{DDQ}

Bits 2:0 **GLOBALVREFINSEL[2:0]**:
[1:0] controls the mode of the PHY V_{REF} DAC and the BP_VREF pin.
00: PHY V_{REF} DAC range0 -- BP_VREF = Hi-Z
01: Reserved
10: PHY V_{REF} DAC range0 -- BP_VREF connected to PLL analog bus
11: PHY V_{REF} DAC range0 -- BP_VREF connected to PHY V_{REF} DAC
[2] controls the DAC range for internal V_{REF} generation.

14.5.121 DDRPHYC maps dfi_rddata_cs_n to destination dimm timing group register (DDRPHYC_MASTER0_DFIWRDATACSDESTMAP)

Address offset: 0x802D0

Reset value: 0x0000 00E4

This register must be default values in LPDDR4 mode Maps dfi_wrdata_cs_n (write chip-select asserted low) to destination dimm timing group. The full dfi interface has 72 values of dfi_wrdata_cs_n, with eight per byte. Byte db (0..8) uses values dfi_wrdata_cs_n[(db*8 + 7):(db*8 + 0)], and this register, which is per byte, to determine the

timing-control registers to use for the asserted chip-select. For simplicity, the subfield description does not include the byte offset for dfi_wrdata_cs_n.

Note: The example values below are the default/reset values. If this register is written to all zeroes, then all writes use register Tx[Dq,Dqs]Dlyd0 timing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIWRDESTM3[1:0]		DFIWRDESTM2[1:0]		DFIWRDESTM1[1:0]		DFIWRDESTM0[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **DFIWRDESTM3[1:0]**:
Maps dfi_wrdata_cs_n[3] to destination DfiWrDestm3 timing (use register Tx[Dq,Dqs]DlyTg3)
For example, if 3 dfi_wrdata_cs_n[0] uses register Tx[Dq,Dqs]DlyTg3 timing.

Bits 5:4 **DFIWRDESTM2[1:0]**:
Maps dfi_wrdata_cs_n[2] to destination DfiWrDestm2 timing
For example, if 2 dfi_wrdata_cs_n[0] uses register Tx[Dq,Dqs]DlyTg2 timing.

Bits 3:2 **DFIWRDESTM1[1:0]**:
Maps dfi_wrdata_cs_n[1] to destination DfiWrDestm1 timing
For example, if 1 dfi_wrdata_cs_n[0] uses register Tx[Dq,Dqs]DlyTg1 timing.

Bits 1:0 **DFIWRDESTM0[1:0]**:
Maps dfi_wrdata_cs_n[0] to destination DfiWrDestm0 timing
For example, if 0 dfi_wrdata_cs_n[0] uses register Tx[Dq,Dqs]DlyTg0 timing.

14.5.122 DDRPHYC MasUpdGoodCtr register (DDRPHYC_MASTER0_MASUPDGOODCTR)

Address offset: 0x802D4

Reset value: 0xXXXX XXXX

Counts successful PHY master interface updates (PPT, periodic phase training).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASUPDGOODCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MASUPDGOODCTR[15:0]**:
This bitfield increments whenever the memory controller acknowledges a PHY master interface request (a request for the PHY) to take over the DFI for PPT (as per section 11 of the Preliminary DFI 4.0 Specification Rev. 2). The counter saturates at max value 65535.

14.5.123 DDRPHYC PhyUpd0GoodCtr register (DDRPHYC_MASTER0_PHYUPD0GOODCTR)

Address offset: 0x802D8

Reset value: 0xXXXX XXXX

Counts successful PHY-initiated DFI0 interface updates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYUPD0GOODCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PHYUPD0GOODCTR[15:0]**:

This bitfield increments whenever the memory controller acknowledges a PHY-initiated DFI0 interface update request. The counter saturates at max value 65535.

14.5.124 DDRPHYC PhyUpd1GoodCtr register (DDRPHYC_MASTER0_PHYUPD1GOODCTR)

Address offset: 0x802DC

Reset value: 0xXXXX XXXX

Counts successful PHY-initiated DFI1 interface updates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYUPD1GOODCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PHYUPD1GOODCTR[15:0]**:

This bitfield increments whenever the memory controller acknowledges a PHY-initiated DFI1 interface update request. The counter saturates at max value 65535.

14.5.125 DDRPHYC CtlUpd0GoodCtr register (DDRPHYC_MASTER0_CTLUPD0GOODCTR)

Address offset: 0x802E0

Reset value: 0xXXXX XXXX

Counts successful memory controller DFI0 interface updates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTLUPD0GOODCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CTLUPD0GOODCTR[15:0]**:

This bitfield increments whenever the PHY acknowledges a memory controller-initiated DFI0 interface update request. The counter saturates at max value 65535.

14.5.126 DDRPHYC MasUpdFailCtr register (DDRPHYC_MASTER0_MASUPDFAILCTR)

Address offset: 0x802E8

Reset value: 0xXXXX XXXX

Counts unsuccessful PHY master interface updates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASUPDFAILCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MASUPDFAILCTR[15:0]**:

This bitfield increments whenever the PHY asserts a PHY master interface request, but the memory controller does not acknowledge the request within the allowed interval (as per section 11 of the Preliminary DFI 4.0 Specification Rev. 2). The counter saturates at max value 65535.

14.5.127 DDRPHYC PhyUpd0FailCtr register (DDRPHYC_MASTER0_PHYUPD0FAILCTR)

Address offset: 0x802EC

Reset value: 0xXXXX XXXX

Counts unsuccessful PHY-initiated DFI0 interface updates

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYUPD0FAILCTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PHYUPD0FAILCTR[15:0]**:

This bitfield increments whenever the PHY asserts a DFI0 interface update request, but the memory controller does not acknowledge the request within the allowed interval (as per section 11 of the Preliminary DFI 4.0 Specification Rev. 2). The counter saturates at max value 65535.

14.5.128 DDRPHYC performance counter enable register (DDRPHYC_MASTER0_PHYPERFCTRENABLE)

Address offset: 0x802F4

Reset value: 0x0000 0000

Each bit corresponds to one performance counter. Set the bit to enable the counter. Clearing the bit stops and clears the counter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPD1FAILCTL	PHYUPD0FAILCTL	MASUPDFAILCTL	CTLUPD1GOODCTL	CTLUPD0GOODCTL	PHYUPD1GOODCTL	PHYUPD0GOODCTL	MASUPDGOODCTL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **PHYUPD1FAILCTL**: PHYUPD1FAILCTLR enable

Bit 6 **PHYUPD0FAILCTL**: PHYUPD0FAILCTLR enable

Bit 5 **MASUPDFAILCTL**: MASUPDFAILCTLR enable

Bit 4 **CTLUPD1GOODCTL**: CTLUPD1GOODCTLR enable

Bit 3 **CTLUPD0GOODCTL**: CTLUPD0GOODCTLR enable

Bit 2 **PHYUPD1GOODCTL**: PHYUPD1GOODCTLR enable

Bit 1 **PHYUPD0GOODCTL**: PHYUPD0GOODCTLR enable

Bit 0 **MASUPDGOODCTL**: MASUPDGOODCTLR enable

14.5.129 DDRPHYC PllPwrDn register (DDRPHYC_MASTER0_PLLPWRDN)

Address offset: 0x8030C

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLPWRDN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLPWRDN**:

This bit is written by the PIE, and the data in here are overwritten. It places the PLL in power down when asserted.

14.5.130 DDRPHYC PllReset register (DDRPHYC_MASTER0_PLLRESET)

Address offset: 0x80310

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLRESET
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLRESET**: Reserved

14.5.131 DDRPHYC PState dependent PLL control 2 register (DDRPHYC_MASTER0_PLLCTRL2)

Address offset: 0x80314

Reset value: 0x0000 0019

This register is to be programmed by bios or other means to set PLL charge-pump settings in various PStates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLFREQSEL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PLLFREQSEL[4:0]**:
Adjusts the loop parameters to compensate for different VCO bias points, and input/output clock division ratios.

14.5.132 DDRPHYC PLL control 0 register (DDRPHYC_MASTER0_PLLCTRL0)

Address offset: 0x80318

Reset value: 0x0000 2008

This register is written by the PIE, and the data in here are overwritten.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLSPARECTRL0	PLLLOCKPHSEL[1:0]		PLLLOCKCNTSEL	PLLGEARSHIFT	PLLRESERVED10X9[1:0]		PLLSYNCBUSBYP	PLLSYNCBUSFLUSH	PLLSELDFIFREQRATIO	PLLBYPASSMODE	PLLPRESET	PLLOUTBYPEN	PLLX2MODE	PLLBYPSEL	PLLSTANDBY
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PLLSPARECTRL0**: Spare bits for PLL control

Bits 14:13 **PLLLOCKPHSEL[1:0]**: Lock detect phase selection
00: Test mode reduced lock window
01: Default operation phase lock
10: Test mode (mismatch check)
11: Test mode (+80ps window)

Bit 12 **PLLLOCKCNTSEL**: Lock detect counter selection
0: Normal lock detector behavior phase lock
1: Test mode (shorten lock cycle count 16->8)

Bit 11 **PLLGEARSHIFT**: PLL fast re-locking mode
0: Normal mode (default)
1: Fast relock gear

Bits 10:9 **PLLRESERVED10X9[1:0]**: Reserved

Bit 8 **PLLSYNCBUSBYP**:
When asserted, bypasses the Pll SyncPulse, and uses a synchronizer of the same latency.

Bit 7 **PLLSYNCBUSFLUSH**:
Used to flush the syncbus logic of the PLL during PHY initialization or LP3 exit sequence. Must only be toggled when syncbus is not used and PLLOUTBYPEN is asserted.

Bit 6 **PLLSELDFIFREQRATIO**: Reserved

Bit 5 **PLLBYPASSMODE**: PLL bypass clock mux control
If set, BypassPclk input is used as byp_pll1x_in clock for PLL (default 0).

Bit 4 **PLLPRESET**: PLL preset mode

Bit 3 **PLLOUTBYPEN**:
Controls the anti-glitch mux on the pllout_x1x2x4 path.
1: pllout_x1x2x4 = byp_pllin_x1
0: pllout_x1x2x4 = VCO (SCD) (selected by x2_mode)

Bit 2 **PLLX2MODE**:
Connects to x2_mode pins of PLL (pllout_x4x2 output frequency selection).
0: PLOUT_X4X2 output = 4*pllin_x1 frequency
1: PLOUT_X4X2 output = 2*pllin_x1 frequency

Bit 1 **PLLBYPSEL**: Reserved

Bit 0 **PLLSTANDBY**:
Connects directly to standby pin of the PLL. Puts PLL into Standby mode.

14.5.133 DDRPHYC PState dependent PLL control 1 register (DDRPHYC_MASTER0_PLLCTRL1)

Address offset: 0x8031C

Reset value: 0x0000 0020

This register is to be programmed by bios or other means to set PLL charge-pump settings in various PStates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLCPPROPCTRL[3:0]				PLLCPINTCTRL[4:0]				
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:5 **PLLCPPROPCTRL[3:0]**:
Connects directly to cp_prop_cntrl<3:0> of the PLL. Charge pump proportional current control.

Bits 4:0 **PLLCPINTCTRL[4:0]**:
Connects directly to cp_int_cntrl<4:0> in the PLL. Charge pump integrating current control.

14.5.134 DDRPHYC PLL testing control register (DDRPHYC_MASTER0_PLLTST)

Address offset: 0x80320

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLDIGTSTSEL[3:0]				PLLANATSTSEL[3:0]				PLLANATSTEN
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:5 **PLLDIGTSTSEL[3:0]**: Digital test mux select

Connects directly to pll_dig_test_sel<2:0> of PLL. Control bits are decoded to enable various digital test signals to be brought out via digital test pin (decode details below).
- PllDigTstOut[1] is observable on BP_ALERT and digital observation pin via MtestMuxSel logic.
- PllDigTstOut[0] is observable on digital observation pin via MtestMuxSel logic only (if enabled in this configuration).
- During mission mode PLLDIGTSTSEL must be set to 0x0.
- Byp = PLLBYPASSMODE value
- Sel = PLLDIGTSTSEL value

Byp | Sel | PllDigTstOut[1] | PllDigTstOut[0] | Description
- 0 | 000 | 0 | 0 | This code must be set during normal use to minimize clock jitter.
- 0 | 001 | fbk_clk_ldet | pllin_x1 | Buffered version of fbk_clk at PFD input
- 0 | 010 | ref_clk_ldet | pllin_x1 | Buffered version of ref_clk at PFD input
- 0 | 011 | pllin_x1 | pllin_x1 | For calibrating delay skew between two test outputs - input reference clock
- 0 | 100 | clk_fbk | pllin_x1 | Buffered version of pllout_x1
- 0 | 101 | pllout_amux_x1x2x4 | pllin_x1 | Syncbus clock output (Fout < Fmax_dto only)
- 0 | 110 | en_count | pll_lock | Lock detector state test
- 0 | 111 | standby_eff | eoc | Calibration state test
- 1 | 000 | 0 | 0 | This code must be set during normal use to minimize clock jitter.
- 1 | 001 | fbk_clk_ldet | pllin_x1 | Buffered version of fbk_clk at PFD input if PLL in mission/byp 1 modes
- 1 | 010 | ref_clk_ldet | pllin_x1 | Buffered version of ref_clk at PFD input -if PLL in mission/byp 1 modes
- 1 | 011 | pllin_x1 | pllin_x1 | For calibrating delay skew between two test outputs - input reference clock
- 1 | 100 | clk_fbk | pllin_x1 | Buffered version of pllout_x1 if PLL in mission/byp 1 modes
- 1 | 101 | pllout_amux_x1x2x4 | pllin_x1 | Syncbus clock output (Fout < Fmax_dto only)
- 1 | 110 | en_count | pll_lock | Lock detector test if PLL in mission/byp 1 modes
- 1 | 111 | standby_eff | eoc | Calibration state test if PLL in mission/byp 1 modes

Note: Fmax_dto=400 MHz

Bits 4:1 **PLLANATSTSEL[3:0]**:

Connects directly to pll_ana_test_sel<3:0> of PLL. Control bits are decoded to enable various analog test signals to be brought out via analog test pin (pll_analog_test) (decode details below).

PLLANATSTSEL: Pll Analog Pin Function: Description

0000: Hi-Z: This code must be set when the PLL is in normal use or when the digital test port is being used for accurate timing measurements.

0001: core: Override vmarg_ext with analog pin.

0010: core: Override regulator input with analog pin.

0011: core: Observe gd voltage on analog test pin.

0100: core: Observe vp voltage on analog test pin.

0101: core: Observe vph voltage on analog test pin.

0110: core: Observe vreg_cp voltage on analog test pin.

0111: core: Observe v2i current (~10uA) on analog test pin.

1000: core: Observe vreg_v2i voltage on analog test pin.

1001: core: Observe vbp voltage from current source on analog pin.

1010: core: Observe vbn voltage from current source on analog pin.

1011: core: NC

1100: core: NC

1101: VSS: 0v probe IR drop test

1110: VP: Core supply probe SCD IR drop test

1111: VPH: Core supply probe Refdiv IR drop test

Bit 0 **PLLANATSTEN**: Analog test port enable

Connects directly to pll_ana_test_en of PLL..

0: Analog test port disabled (hi-Z)

1: Analog test port enabled

14.5.135 DDRPHYC PllLockStatus register (DDRPHYC_MASTER0_PLLLOCKSTATUS)

Address offset: 0x80324

Reset value: 0xXXXX XXXX

PLL pll_lock pin output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLLOCKSTATUS
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLLOCKSTATUS**: Directly connected to the pll_Lock output

When asserted, PLL macro has asserted the lock status. To avoid metastability issues, this bit must perform a majority-vote after reading this bit several times.

14.5.136 DDRPHYC PII TestMode register (DDRPHYC_MASTER0_PLLTESTMODE)

Address offset: 0x80328

Reset value: 0x0000 0124

Additional controls for PLL CP/VCO modes of operation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLTESTMODE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PLLTESTMODE[15:0]**:

Default values for this bitfield must be used unless directed otherwise by Synopsys.
Directly connected to testmode[15:0] pin of PLL

14.5.137 DDRPHYC PLL control 3 register (DDRPHYC_MASTER0_PLLCTRL3)

Address offset: 0x8032C

Reset value: 0x0000 01F0

It is recommended to use default values for this register. Otherwise, it can be programmed by bios or other means.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLENCAL	PLLFORCECAL	PLLDACVALIN[4:0]					PLLMAXRANGE[4:0]					PLLSPARE[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PLLENCAL**:

0: The calibration does not run.
1: The calibration runs at standby rising edge.

Bit 14 **PLLFORCECAL**:

Connects directly to force_cal of PLL.
0: Previous stored value used
1: Forces the calibration code dacval_in<4:0> to be used at preset mode.

Bits 13:9 **PLLDACVALIN[4:0]**:

Connects directly to dacval_in<4:0> of PLL. If this bitfield is set, the calibration DAC input current setting can be forced at rising edge of preset.

Bits 8:4 **PLLMAXRANGE[4:0]**:
Connects directly to maxrange of PLL. Setting for saturation control of PLL.

Bits 3:0 **PLLSPARE[3:0]**: Spare bits

14.5.138 DDRPHYC PState dependent PLL control 4 register (DDRPHYC_MASTER0_PLLCTRL4)

Address offset: 0x80330

Reset value: 0x0000 017F

This register is to be programmed by bios or other means to set PLL charge-pump settings in various PStates.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLCPPROPGSCTRL[3:0]				PLLCPINTGSCTRL[4:0]				
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:5 **PLLCPPROPGSCTRL[3:0]**:
Connects directly to cp_prop_gs_cntrl<3:0> of PLL. Charge pump proportional current control for fast relock and gearshift.

Bits 4:0 **PLLCPINTGSCTRL[4:0]**:
Connects directly to cp_int_gs_cntrl<4:0> in PLL. Charge pump integrating current control for fast relock and gearshift.

14.5.139 DDRPHYC PllEndofCal register (DDRPHYC_MASTER0_PLLENDOFCAL)

Address offset: 0x80334

Reset value: 0xXXXX XXXX

PLL eoc (end of calibration) output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLENDOFCAL
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLENDOFCAL**:
Directly connected to the PLL eoc output.

14.5.140 DDRPHYC PllStandbyEff register (DDRPHYC_MASTER0_PLLSTANDBYEFF)

Address offset: 0x80338

Reset value: 0xXXXX XXXX

PLL standby_eff (effective standby) output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLSTANDBYEFF
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PLLSTANDBYEFF**: State off PLL standby (in PHY LP2 states)

14.5.141 DDRPHYC PllDacValOut register (DDRPHYC_MASTER0_PLLDACVALOUT)

Address offset: 0x8033C

Reset value: 0xXXXX XXXX

PLL dacval_out output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLDACVALOUT[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PLLDACVALOUT[4:0]**:

Directly connected to the PLL dacval_out output. Used for observation of PLL DAC code

14.5.142 DDRPHYC LcdlDbgCntl register (DDRPHYC_MASTER0_LCDLDBGCNTL)

Address offset: 0x8038C

Reset value: 0x0000 0000

Controls for use in observing and testing the LCDLs.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCDLSTATUSSEL[3:0]				LCDLTSTENABLE	LCDLFINESNAP	LCDLFINEOVR	LCDLFINEOVRVAL[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **LCDLSTATUSSEL[3:0]**: Selection of the LCDL status

Selects the LCDL status from among the status for the 16 LCDLs in the DBYTE, for reading via DXLCDLSTATUS. Selects an LCDL from among the LCDLs in the ANIB for reading via ACLCDLSTATUS.

1111: lcdl_rxclk1t reserved
1110: lcdl_rxclk0t reserved
1101: lcdl_rxclk1c reserved
1100: lcdl_rxclk0c reserved
1011: lcdl_rxen1 anib11-tx
1010: lcdl_rxen0 anib10-tx
1001: lcdl_txln9 (dqs-lower) anib9-tx
1000: lcdl_txln8 (dm/dqs-upper) anib8-tx
0111: lcdl_txln7 (dq7) anib7-tx
0110: lcdl_txln6 (dq6) anib6-tx
0101: lcdl_txln5 (dq5) anib5-tx
0100: lcdl_txln4 (dq4) anib4-tx
0011: lcdl_txln3 (dq3) anib3-tx
0010: lcdl_txln2 (dq2) anib2-tx
0001: lcdl_txln1 (dq1) anib1-tx
0000: lcdl_txln0 (dq0) anib0-tx

Bit 11 **LCDLTSTENABLE**: Debug/test operations enable

Ovr, Snap, StickyLock, StickyUnlock, and LiveLock status

Bit 10 **LCDLFINESNAP**: Latch enable for reading the present LCDL 1UI estimate code in LCDLFINESNAPVAL, and the present phase-detector value in LCDLPHDSNAPVAL.

Bit 9 **LCDLFINEOVR**:

Forces the value of the present LCDL 1UI estimate code to be LCDLFINEOVRVAL for all LCDLs.

Bits 8:0 **LCDLFINEOVRVAL[8:0]**: Value forced as the initial value while LCDLTSTENABLE = 1 and LCDLFINEOVR

After the deassertion of LCDLFINEOVR, the locking state-machine attempts locking.

14.5.143 DDRPHYC Debug status of the DBYTE LCDL register (DDRPHYC_MASTER0_ACLCDLSTATUS)

Address offset: 0x80390

Reset value: 0xXXXX XXXX

Status of ANIB macro LCDL selected by LCDLSTATUSSEL

Note: There is just one copy of this register, with the information from the implemented ANIBs multiplexed per LCDLSTATUSSEL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ACLCDLLIVELOCK	ACLCDLSTICKYUNLOCK	ACLCDLSTICKYLOCK	ACLCDLPHDSNAPVAL	ACLCDLFINESNAPVAL[9:0]									
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ACLCDLLIVELOCK**: Present value of whether the LCDL is locked
Valid when LCDLTSTENABLE = 1.

Bit 12 **ACLCDLSTICKYUNLOCK**: Latched value of whether the LCDL ever lost lock after the assertion of LCDLTSTENABLE.

Bit 11 **ACLCDLSTICKYLOCK**: Latched value of whether the LCDL ever achieved lock after the assertion of LCDLTSTENABLE.

Bit 10 **ACLCDLPHDSNAPVAL**: Value of the LCDL phase-detector output, latched by pulse on LCDLFINESNAP, while LCDLTSTENABLE = 1.

Bits 9:0 **ACLCDLFINESNAPVAL[9:0]**: Value of the LCDL 1UI estimate code, latched by pulse on LCDLFINESNAP, while LCDLTSTENABLE = 1. Index 9 is reserved for growth.

14.5.144 DDRPHYC CUSTPHYREV register (DDRPHYC_MASTER0_CUSTPHYREV)

Address offset: 0x803B4

Reset value: 0xXXXX XXXX

Customer settable PHY version number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUSTPHYREV[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CUSTPHYREV[5:0]**: Customer settable PHY version number
The value is derived from the define DWC_DDRPHY_CUST_PHYREV.

14.5.145 DDRPHYC PHYREV register (DDRPHYC_MASTER0_PHYREV)

Address offset: 0x803B8

Reset value: 0x0000 0100

Hardware version of this PHY, excluding the PUB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYMJR[7:0]								PHYMDR[3:0]				PHYMNR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **PHYMJR[7:0]**: PHY major revision

Bits 7:4 **PHYMDR[3:0]**: PHY moderate revision

Bits 3:0 **PHYMNR[3:0]**: PHY minor update

14.5.146 DDRPHYC Start vector value for LP3-exit or init PIE sequence register (DDRPHYC_MASTER0_LP3EXITSEQ0BSTARTVECTOR)

Address offset: 0x803BC

Reset value: 0x0000 00DE

Programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes. Set the start vector to be used in LP3-exit or Init PIE sequence.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LP3EXITSEQ0BSTARTVECPLLBYPASSED[3:0]				LP3EXITSEQ0BSTARTVECPLLENABLED[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **LP3EXITSEQ0BSTARTVECPLLBYPASSED[3:0]**: PIE start vector value to be used for LP3 exit or init and target P-state has PLL bypassed

Bits 3:0 **LP3EXITSEQ0BSTARTVECPLLENABLED[3:0]**: PIE start vector value to be used for LP3 exit or init and target P-state has PLL enabled

14.5.147 DDRPHYC DFI frequency translation 0 register (DDRPHYC_MASTER0_DFIFREQXLAT0)

Address offset: 0x803C0

Reset value: 0x0000 0000

DFI frequency translation register 0, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL3[3:0]				DFIFREQXLATVAL2[3:0]				DFIFREQXLATVAL1[3:0]				DFIFREQXLATVAL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL3[3:0]**: Sequencer start vector used when dfi_freq value is 3

Bits 11:8 **DFIFREQXLATVAL2[3:0]**: Sequencer start vector used when dfi_freq value is 2

Bits 7:4 **DFIFREQXLATVAL1[3:0]**: Sequencer start vector used when dfi_freq value is 1

Bits 3:0 **DFIFREQXLATVAL0[3:0]**: Sequencer start vector used when dfi_freq value is 0

14.5.148 DDRPHYC DFI frequency translation 1 register (DDRPHYC_MASTER0_DFIFREQXLAT1)

Address offset: 0x803C4

Reset value: 0x0000 0000

DFI frequency translation register 1, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL7[3:0]				DFIFREQXLATVAL6[3:0]				DFIFREQXLATVAL5[3:0]				DFIFREQXLATVAL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL7[3:0]**: Sequencer start vector used when dfi_freq value is 7

Bits 11:8 **DFIFREQXLATVAL6[3:0]**: Sequencer start vector used when dfi_freq value is 6

Bits 7:4 **DFIFREQXLATVAL5[3:0]**: Sequencer start vector used when dfi_freq value is 5

Bits 3:0 **DFIFREQXLATVAL4[3:0]**: Sequencer start vector used when dfi_freq value is 4

14.5.149 DDRPHYC DFI frequency translation 2 register (DDRPHYC_MASTER0_DFIFREQXLAT2)

Address offset: 0x803C8

Reset value: 0x0000 0000

DFI frequency translation register 2, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL11[3:0]				DFIFREQXLATVAL10[3:0]				DFIFREQXLATVAL9[3:0]				DFIFREQXLATVAL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL11[3:0]**: Sequencer start vector used when dfi_freq value is 11

Bits 11:8 **DFIFREQXLATVAL10[3:0]**: Sequencer start vector used when dfi_freq value is 10

Bits 7:4 **DFIFREQXLATVAL9[3:0]**: Sequencer start vector used when dfi_freq value is 9

Bits 3:0 **DFIFREQXLATVAL8[3:0]**: Sequencer start vector used when dfi_freq value is 8

14.5.150 DDRPHYC DFI frequency translation 3 register (DDRPHYC_MASTER0_DFIFREQXLAT3)

Address offset: 0x803CC

Reset value: 0x0000 0000

DFI frequency translation register 3, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL15[3:0]				DFIFREQXLATVAL14[3:0]				DFIFREQXLATVAL13[3:0]				DFIFREQXLATVAL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL15[3:0]**: Sequencer start vector used when dfi_freq value is 15

Bits 11:8 **DFIFREQXLATVAL14[3:0]**: Sequencer start vector used when dfi_freq value is 14

Bits 7:4 **DFIFREQXLATVAL13[3:0]**: Sequencer start vector used when dfi_freq value is 13

Bits 3:0 **DFIFREQXLATVAL12[3:0]**: Sequencer start vector used when dfi_freq value is 12

14.5.151 DDRPHYC DFI frequency translation 4 register (DDRPHYC_MASTER0_DFIFREQXLAT4)

Address offset: 0x803D0

Reset value: 0x0000 0000

DFI frequency translation register 4, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL19[3:0]				DFIFREQXLATVAL18[3:0]				DFIFREQXLATVAL17[3:0]				DFIFREQXLATVAL16[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL19[3:0]**: Sequencer start vector used when dfi_freq value is 19

Bits 11:8 **DFIFREQXLATVAL18[3:0]**: Sequencer start vector used when dfi_freq value is 18

Bits 7:4 **DFIFREQXLATVAL17[3:0]:** Sequencer start vector used when dfi_freq value is 17

Bits 3:0 **DFIFREQXLATVAL16[3:0]**: Sequencer start vector used when dfi_freq value is 16

14.5.152 DDRPHYC DFI frequency translation 5 register (DDRPHYC_MASTER0_DFIFREQXLAT5)

Address offset: 0x803D4

Reset value: 0x0000 0000

DFI frequency translation register 5, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL23[3:0]				DFIFREQXLATVAL22[3:0]				DFIFREQXLATVAL21[3:0]				DFIFREQXLATVAL20[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL23[3:0]**: Sequencer start vector used when dfi_freq value is 23

Bits 11:8 **DFIFREQXLATVAL22[3:0]**: Sequencer start vector used when dfi_freq value is 22

Bits 7:4 **DFIFREQXLATVAL21[3:0]**: Sequencer start vector used when dfi_freq value is 21

Bits 3:0 **DFIFREQXLATVAL20[3:0]**: Sequencer start vector used when dfi_freq value is 20

14.5.153 DDRPHYC DFI frequency translation 6 register (DDRPHYC_MASTER0_DFIFREQXLAT6)

Address offset: 0x803D8

Reset value: 0x0000 0000

DFI frequency translation register 6, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL27[3:0]				DFIFREQXLATVAL26[3:0]				DFIFREQXLATVAL25[3:0]				DFIFREQXLATVAL24[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL27[3:0]**: Sequencer start vector used when dfi_freq value is 27

Bits 11:8 **DFIFREQXLATVAL26[3:0]**: Sequencer start vector used when dfi_freq value is 26

Bits 7:4 **DFIFREQXLATVAL25[3:0]**: Sequencer start vector used when dfi_freq value is 25

Bits 3:0 **DFIFREQXLATVAL24[3:0]**: Sequencer start vector used when dfi_freq value is 24

14.5.154 DDRPHYC DFI frequency translation 7 register (DDRPHYC_MASTER0_DFIFREQXLAT7)

Address offset: 0x803DC

Reset value: 0x0000 0000

DFI frequency translation register 7, programmed by `dwc_ddrphy_phyinit_C_initPhyConfig()` to support frequency changes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFIFREQXLATVAL31[3:0]				DFIFREQXLATVAL30[3:0]				DFIFREQXLATVAL29[3:0]				DFIFREQXLATVAL28[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **DFIFREQXLATVAL31[3:0]**: Sequencer start vector used when dfi_freq value is 31

Bits 11:8 **DFIFREQXLATVAL30[3:0]**: Sequencer start vector used when dfi_freq value is 30

Bits 7:4 **DFIFREQXLATVAL29[3:0]**: Sequencer start vector used when dfi_freq value is 29

Bits 3:0 **DFIFREQXLATVAL28[3:0]**: Sequencer start vector used when dfi_freq value is 28

14.5.155 DDRPHYC TxRdPtrInit register (DDRPHYC_MASTER0_TXRDPTRINIT)

Address offset: 0x803E0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXRDP TRINIT
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TXRDPTRINIT**:

This bit controls TxRdPtrInit, and is meant to be written by the PState sequencer as part of the power state switching sequence. It must be set once the transition to LP2 is complete, and as follows:

- 32 cycles before the deasserting of init_complete. It must be cleared once the transition to an active PState is complete.
- 32 cycles before the assertion of init_complete. It must be reset to 1 on both warm and cold resets.

14.5.156 DDRPHYC DfiInitComplete register (DDRPHYC_MASTER0_DFIINITCOMPLETE)

Address offset: 0x803E4

Reset value: 0x0000 0000

DFI init complete control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIINIT COMP LETE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DFIINITCOMPLETE**:

This bit directly controls DfiInitComplete, and is meant to be written by the PState sequencer as part of the power state switching sequence. It must set once the transition to LP2 is complete. It must be cleared once the transition to an active PState is complete. It must be reset to a 0 on both warm and cold resets.

14.5.157 DDRPHYC DfiFreqRatio register (DDRPHYC_MASTER0_DFIFREQRATIO)

Address offset: 0x803E8

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQRATIO[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **DFIFREQRATIO[1:0]**: Ratio DfiCtlClk:MemClk

00 = 1:1 mode

01 = 1:2 mode

1x = 1:4 mode

When DWC_DDRPHY_FIXED_DFICTLCLK_RATIO is defined, this bitfield is ignored and 1:2 mode is selected.

14.5.158 DDRPHYC RxFifoCheck register (DDRPHYC_MASTER0_RXFIFOCHECKS)

Address offset: 0x803EC

Reset value: 0x0000 0000

Enable more frequent consistency checks of the Rx FIFOs.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOFREQUENTRXFIFOCHECKS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DOFREQUENTRXFIFOCHECKS**:

0: Read data FIFO pointer consistency checks are performed only during sideband transactions (phyupd, ctrlupd, lp_data, phymstr), or when dfi_init_start is asserted. This option relies upon intentional quiescing of the DFI interface in order for the check to be performed.

1: Read data FIFO consistency checks are performed as when this bit is 0, and also at the beginning of each break in read data traffic from the DRAM. This options provides the lowest latency in reporting of a discrepancy in the read data FIFO pointer values.

14.5.159 DDRPHYC MTestCombo output enable on core-side copy register (DDRPHYC_MASTER0_MTESTDTOCTRL)

Address offset: 0x803FC

Reset value: 0x0000 0000

If set, this register allows the MTestCombo observability signal to be routed out core-side through the dwc_ddrphy_dto pin

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTEST DTOEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MTESTDTOEN**:

This bit is programmed as follows:
- MTESTdtoEn==[0], dwc_ddrphy_dto is squelched (0).
- MTESTdtoEn==[1], dwc_ddrphy_dto reflects the observability signal multiplexed on MTestCombo.

14.5.160 DDRPHYC MapCAA[0to7]toDfi register (DDRPHYC_MASTER0_MAPCAAxTODFI)

Address offset: 0x80400 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000, 0x0000 0001, 0x0000 0002, 0x0000 0003, 0x0000 0004, 0x0000 0005, 0x0000 0006, 0x0000 0007

Maps PHY CAA lane x from dfi0_address of the index of the register contents.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAATODFI[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **MAPCAATODFI[3:0]**:

For LPDDR4 application, this bitfield maps a dfi0_address to CAA x. It is not used for DDR3 and DDR4 applications. Each bit must have a unique value within the set.

For example, if PHY CAA0 is mapped from dfi0_address[2], then MAPCAATODFI must be 2.

- LPDDR4 ACX3, ACX6, ACX10, ACX12 registers MAPCAA[0..5]toDfi are used.
  - Values 0-5 are valid.
  - Values 6-15 are reserved.
  - MAPCAA[6..7]toDfi are reserved.

14.5.161 DDRPHYC interrupt enable register (DDRPHYC_MASTER0_PHYINTERRUPTENABLE)

Address offset: 0x8046C

Reset value: 0x0000 0000

Enables various interrupts to appear in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS register. Only interrupts which are enabled by setting the appropriate bits in this register are observable in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS. If a 0 is written in a particular bit, the corresponding interrupt is not visible in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYHWRESERVEDEN[4:0]					PHYRXFIFOCHECKEN	PHYVTDRIFTALARMEN[1:0]		PHYFWRESERVEDEN[4:0]					PHYTRNGFAILEN	PHYINITCMPLTEN	PHYTRNGCMPLTEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **PHYHWRESERVEDEN[4:0]**: Reserved

Bit 10 **PHYRXFIFOCHECKEN**: Rx FIFO pointer check interrupt enable
0: Interrupt not enabled
1: Interrupt enabled

Bits 9:8 **PHYVTDRIFTALARMEN[1:0]**: PHY VT drift alarm interrupt enable
0: Interrupt not enabled
1: Interrupt enabled

Bits 7:3 **PHYFWRESERVEDEN[4:0]**: Reserved

Bit 2 **PHYTRNGFAILEN**: PHY training failure interrupt enable
0: Interrupt not enabled
1: Interrupt enabled

Bit 1 **PHYINITCMPLTEN**: PHY initialization complete interrupt enable
0: Interrupt not enabled
1: Interrupt enabled

Bit 0 **PHYTRNGCMPLTEN**: PHY training complete interrupt enable
0: Interrupt not enabled
1: Interrupt enabled

14.5.162 DDRPHYC interrupt firmware control register (DDRPHYC_MASTER0_PHYINTERRUPTFWCONTROL)

Address offset: 0x80470

Reset value: 0x0000 0000

A 0-to-1 transition for any of the interrupts sets the corresponding bit location in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS, if the corresponding bit location in DDRPHYC_MASTER0_PHYINTERRUPTENABLE is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYFWRESERVEDFW[4:0]					PHYTRNGFAILFW	PHYINITCMPLTFW	PHYTRNGCMPLTFW
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PHYFWRESERVEDFW[4:0]**: Reserved
00000: Interrupt not asserted
00001: Interrupt asserted
Others: Reserved

Bit 2 **PHYTRNGFAILFW**: PHY training failure firmware interrupt
Written only by PHY firmware.
0: Interrupt not asserted
1: Interrupt asserted

Bit 1 **PHYINITCMPLTFW**: PHY initialization complete firmware interrupt
The PhyInitCmpIt interrupt is not written by PHY firmware. It can be written by the user once dfi_init_complete is asserted.
0: Interrupt not asserted
1: Interrupt asserted

Bit 0 **PHYTRNGCMPLTFW**: PHY training complete firmware interrupt
Written only by PHY firmware.
0: Interrupt not asserted
1: Interrupt asserted

14.5.163 DDRPHYC interrupt mask register (DDRPHYC_MASTER0_PHYINTERRUPTMASK)

Address offset: 0x80474

Reset value: 0x0000 0000

This register is used to mask various interrupts to prevent them from affecting the output pin dwc_ddrphy_int_n. Only interrupts which are not masked by clearing to 0 the corresponding bit in this register are able to affect the dwc_ddrphy_int_n output when asserted in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS. If a bit is set, the interrupt is still visible

in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS but this interrupt does not affect the output pin dwc_ddrphy_int_n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYHWRESERVEDMSK[4:0]					PHYRXFIFOCHECKMSK	PHYVTDRIFTALARMMSK[1:0]		PHYFWRESERVEDMSK[4:0]					PHYTRNGFAILMSK	PHYINITCMPLTMSK	PHYTRNGCMPLTMSK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **PHYHWRESERVEDMSK[4:0]**: Reserved

Bit 10 **PHYRXFIFOCHECKMSK**: Mask for the Rx FIFO pointers check interrupt
0: Interrupt not masked
1: Interrupt masked

Bits 9:8 **PHYVTDRIFTALARMMSK[1:0]**: Mask for the PHY VT drift alarm interrupts
0: Interrupt not masked
1: Interrupt masked

Bits 7:3 **PHYFWRESERVEDMSK[4:0]**: Reserved

Bit 2 **PHYTRNGFAILMSK**: Mask for the PHY training failure interrupt
0: Interrupt not masked
1: Interrupt masked

Bit 1 **PHYINITCMPLTMSK**: Mask for the PHY initialization complete interrupt
0: Interrupt not masked
1: Interrupt masked

Bit 0 **PHYTRNGCMPLTMSK**: Mask for the PHY training complete interrupt
0: Interrupt not masked
1: Interrupt masked

14.5.164 DDRPHYC interrupt clear register (DDRPHYC_MASTER0_PHYINTERRUPTCLEAR)

Address offset: 0x80478

Reset value: 0x0000 0000

This register clears various interrupts, and negates the output pin dwc_ddrphy_int_n if the cleared interrupt is the cause of the assertion of the output pin dwc_ddrphy_int_n. A 0-to-1

transition on a bit clears the corresponding interrupt, and causes the hardware to immediately clear the bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYHWRESERVEDCLR[4:0]					PHYRXFIFOCHECKCLR	PHYVTDRIFTALARMCLR[1:0]		PHYFWRESERVEDCLR[4:0]					PHYTRNGFAILCLR	PHYINITCMPLTCLR	PHYTRNGCMPLTCLR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **PHYHWRESERVEDCLR[4:0]**: Reserved

Bit 10 **PHYRXFIFOCHECKCLR**: Rx FIFO pointers check interrupt clear
0: Interrupt not affected
1: Interrupt cleared

Bits 9:8 **PHYVTDRIFTALARMCLR[1:0]**: PHY VT drift alarm interrupt clear
0: Interrupt not affected
1: Interrupt cleared

Bits 7:3 **PHYFWRESERVEDCLR[4:0]**: Reserved

Bit 2 **PHYTRNGFAILCLR**: PHY training failure interrupt clear
0: Interrupt not affected
1: Interrupt cleared

Bit 1 **PHYINITCMPLTCLR**: PHY initialization complete interrupt clear
0: Interrupt not affected
1: Interrupt cleared

Bit 0 **PHYTRNGCMPLTCLR**: PHY training complete interrupt clear
0: Interrupt not affected
1: Interrupt cleared

14.5.165 DDRPHYC interrupt status register (DDRPHYC_MASTER0_PHYINTERRUPTSTATUS)

Address offset: 0x8047C

Reset value: 0xXXXX XXXX

This read-only register displays the current status of various PHY interrupts. The PHY is the only entity which can write to it. In order for a bit to become set in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS, the corresponding bit must be set in DDRPHYC_MASTER0_PHYINTERRUPTENABLE.

If an interrupt is set in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS and the corresponding bit in DDRPHYC_MASTER0_PHYINTERRUPTENABLE is cleared, the dwc_ddrphy_int_n output signal is asserted. Once the dwc_ddrphy_int_n output signal is asserted, it can only be negated when all unmasked interrupts in

DDRPHYC_MASTER0_PHYINTERRUPTSTATUS have been cleared by setting the corresponding bit in DDRPHYC_MASTER0_PHYINTERRUPTSTATUS.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYHWRESERVED[4:0]					PHYRXFIFOCHECK	VTDRIFTALARM[1:0]		PHYFWRESERVED[4:0]					PHYTRNGFAIL	PHYINITCMPLT	PHYTRNGCMPLT
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **PHYHWRESERVED[4:0]**: Reserved

Bit 10 **PHYRXFIFOCHECK**:

A mechanism in the PHY checks the read FIFO pointers for consistency at times they are idle. If a check fails, this interrupt is asserted, and detailed status is in DDRPHYC_DBYTE0_RXFIFOCHECKSTATUS.

0: Interrupt not enabled and asserted

1: Interrupt enabled and asserted

Bits 9:8 **VTDRIFTALARM[1:0]**: PHY VT drift alarm interrupt

Assertion of this interrupt indicates the number of tap points in the master DLL has shifted by an amount greater than 2 * DFIPHYUPDINTTHRESHOLD. There is one bit for each master DLL, and they are fully independent of one another. See DFIPHYUPDINTTHRESHOLD[3:0] for additional information.

0: Interrupt not enabled and asserted

1: Interrupt enabled and asserted

Bits 7:3 **PHYFWRESERVED[4:0]**: Reserved

Bit 2 **PHYTRNGFAIL**: PHY training failure interrupt

Assertion of this interrupt indicates there was an issue during training which did not allow the PHY to successfully complete the training.

0: Interrupt not enabled and asserted

1: Interrupt enabled and asserted

Bit 1 **PHYINITCMPLT**: PHY initialization complete interrupt

Written by the user. Assertion of this interrupt is meant to indicate that the PIE has been fully loaded, and dfi_init_complete has been asserted.

0: Interrupt not enabled and asserted

1: Interrupt enabled and asserted

Bit 0 **PHYTRNGCMPLT**: PHY training complete interrupt

Assertion of this interrupt indicates that the PHY has successfully trained the memory interface at all indicated frequencies, and is ready for the PIE to be loaded.

0: Interrupt not enabled and asserted

1: Interrupt enabled and asserted

14.5.166 DDRPHYC HwtSwizzleHwtAddress0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESSx)

Address offset: 0x80480 + 0x4 * x (x = 0 to 17)

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTADDRESS[4:0]**:

This bitfield is used in DDR3/DDR4 mode where a user has remapped the DFI inputs to the PHY. It does not affect the mapping of these signals, or remaps the tristate properties of the bumps. However, when the user remaps the DFI signals, the HWT inputs must be mapped in a similar fashion using this bitfield.

Each remappable HWT has a bit associated with it. When the bit value is zero, the HWT is not remapped. When the register value is non-zero, the signal associated with the number below is mapped to the signal associated with the bit.

For example, if the value for the register associated with hwt_act_n (HwtSwizzleHwtActN) is set to 27, then hwt_parity_in is mapped to hwt_act_n.

These bits must always be 0 for LPDDR4. Some bits are used only for DDR3 or DDR4 (noted in the descriptions).

0: No remapping
1: hwt_address[0]
2: hwt_address[1]
3: hwt_address[2]
4: hwt_address[3]
5: hwt_address[4]
6: hwt_address[5]
7: hwt_address[6]
8: hwt_address[7]
9: hwt_address[8]
10: hwt_address[9]
11: hwt_address[10]
12: hwt_address[11]
13: hwt_address[12]
14: hwt_address[13]
15: hwt_address[14] (DDR3 only)
16: hwt_address[15] (DDR3 only)
17: hwt_address[17] (DDR4 only)
18: hwt_act_n (DDR4 only)
19: hwt_bank[0]
20: hwt_bank[1]
21: hwt_bank[2] (DDR3 only)
22: hwt_bg[0] (DDR4 only)
23: hwt_bg[1] (DDR4 only)
24: hwt_cas_n
25: hwt_ras_n
26: hwt_we_n
27: hwt_parity_in

14.5.167 DDRPHYC HwtSwizzleHwtActN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTACTN)

Address offset: 0x804C4

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTACTN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTACTN[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details. Used only for DDR4.

14.5.168 DDRPHYC HwtSwizzleHwtBank0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK0)

Address offset: 0x804C8

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK0[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTBANK0[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.169 DDRPHYC HwtSwizzleHwtBank1 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK1)

Address offset: 0x804CC

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTBANK1[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.170 DDRPHYC HwtSwizzleHwtBank2 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK2)

Address offset: 0x804D0

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK2[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTBANK2[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details. Used only for DDR3.

14.5.171 DDRPHYC HwtSwizzleHwtBg0 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBG0)

Address offset: 0x804D4

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBG0[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTBG0[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details. Used only for DDR4.

14.5.172 DDRPHYC HwtSwizzleHwtBg1 register (DDRPHYC_MASTER0_HWTSWIZZLEHWTBG1)

Address offset: 0x804D8

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBG1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTBG1[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details. Used only for DDR4.

14.5.173 DDRPHYC HwtSwizzleHwtCasN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTCASN)

Address offset: 0x804DC

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTCASN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTCASN[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.174 DDRPHYC HwtSwizzleHwtRasN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTRASN)

Address offset: 0x804E0

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTRASN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTRASN[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.175 DDRPHYC HwtSwizzleHwtWeN register (DDRPHYC_MASTER0_HWTSWIZZLEHWTWEN)

Address offset: 0x804E4

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTWEN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTWEN[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.176 DDRPHYC HwtSwizzleHwtParityIn register (DDRPHYC_MASTER0_HWTSWIZZLEHWTPARITYIN)

Address offset: 0x804E8

Reset value: 0x0000 0000

Signal swizzle selection for HWT swizzle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTPARITYIN[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **HWTSWIZZLEHWTPARITYIN[4:0]**:
See HWTSWIZZLEHWTADDRESS0[4:0] description for details.

14.5.177 DDRPHYC small delays on handshake signals register 0 (DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS0)

Address offset: 0x804F0

Reset value: 0x0000 0000

Add assertion/deassertion delays on handshake signals. Logic assumes that DFI signal assertions exceed the programmed delays

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRLUPDREQDELAY0[3:0]				CTRLUPDACKDELAY0[3:0]				PHYUPDREQDELAY0[3:0]				PHYUPDACKDELAY0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **CTRLUPDREQDELAY0[3:0]**:
Adds 0-15 DfiClks of delay after the PHY completes all PHY update activities, before deasserting dfi0_ctrlupd_ack.

Bits 11:8 **CTRLUPDACKDELAY0[3:0]**:
Adds 0-15 DfiClks of delay after dfi0_ctrlupd_req asserts, before the PHY takes any action.

Bits 7:4 **PHYUPDREQDELAY0[3:0]**:
Adds 0-15 DfiClks of delay after the PHY completes all PHY update activities, before deasserting dfi0_phyupd_req.

Bits 3:0 **PHYUPDACKDELAY0[3:0]**:
Adds 0-15 DfiClks of delay after dfi0_phyupd_ack asserts, before the PHY takes any action (such as starting DDL calibration).

14.5.178 DDRPHYC small delays on handshake signals register 1 (DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS1)

Address offset: 0x804F4

Reset value: 0x0000 0000

Add assertion/deassertion delays on handshake signals. Logic assumes that dfi signal assertions exceed the programmed delays

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPDREQDELAY1[3:0]				PHYUPDACKDELAY1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PHYUPDREQDELAY1[3:0]**:
Adds 0-15 DfiClks of delay after the PHY completes all PHY update activities, before deasserting dfi1_phyupd_req.

Bits 3:0 **PHYUPDACKDELAY1[3:0]**:
Adds 0-15 DfiClks of delay after dfi1_phyupd_ack asserts, before the PHY takes any action (such as starting DDL calibration).

14.5.179 DDRPHYC remote impedance calibration status register (DDRPHYC_MASTER0_REMOTEIMPCAL)

Address offset: 0x804F8

Reset value: 0xXXXX XXXX

This register is written by hardware to provide information about remote impedance calibration status. Its value allows software to determine if the PHY acts as calibration slave (1) or master (0). If the PHY acts as calibration slave, this register can be checked to determine if calibration codes have been captured.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLAVE CODE UPDATED	CALIB SLAVE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SLAVECODEUPDATED**:
- if DWC_DDRPHY_REMOTE_IMP_CAL is defined, this bit is set by hardware when the PHY acts as calibration slave, and calibration codes from the master have been loaded.
- if DWC_DDRPHY_REMOTE_IMP_CAL is not defined, this bit is reserved.

Bit 0 **CALIBSLAVE**:
- if DWC_DDRPHY_REMOTE_IMP_CAL is defined, this bit is set by hardware when the PHY acts as calibration slave.
- if DWC_DDRPHY_REMOTE_IMP_CAL is not defined, this bit is reserved.

14.5.180 DDRPHYC PhyInLP3 register (DDRPHYC_INITENG0_PHYINLP3)

Address offset: 0x2400A0

Reset value: 0x0000 0001

Indicator for PIE lower power 3 (LP3) status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYIN LP3
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PHYINLP3**:

This bit is set by the PIE once the LP3 entry sequence is completed. it is cleared during LP3 exit sequence. System software can read this bit for the status of the PIE.

14.5.181 DDRPHYC Ucclk and Hclk enable register (DDRPHYC_DRTUB0_UCCLKHCLKENABLES)

Address offset: 0x300200

Reset value: 0x0000 0003

Ucclk/Hclk enable. Ucclk and Hclk run at either DFICLK or are gated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HCLKEN	UCCLKEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HCLKEN**:

When the training is complete (and assuming no further need for training hardware), this bit must be cleared to reduce power.

Bit 0 **UCCLKEN**:

When the training is complete (and assuming no further need for the microcontroller), this bit must be cleared to reduce power.

14.5.182 DDRPHYC CurPstate0b register (DDRPHYC_DRTUB0_CURPSTATE0B)

Address offset: 0x300204

Reset value: 0x0000 0000

PIE current PState value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURPSTATE0B[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CURPSTATE0B[3:0]**: PIE current PState

This bitfield is used to select values for writing by the PState sequencer. It is written at the beginning of the PState switch. It can also be used to reference which PState the sequencer thinks is in.

14.5.183 DDRPHYC CUSTPUBREV register (DDRPHYC_DRTUB0_CUSTPUBREV)

Address offset: 0x3003B4

Reset value: 0xXXXX XXXX

Customer settable PUB version number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUSTPUBREV[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CUSTPUBREV[5:0]**: Customer settable PUB version number

14.5.184 DDRPHYC PUBREV register (DDRPHYC_DRTUB0_PUBREV)

Address offset: 0x3003B8

Reset value: 0x0000 2420

Hardware version of this PUB, excluding the PHY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUBMJR[7:0]								PUBMDR[3:0]				PUBMNR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **PUBMJR[7:0]**: PUB major revision

Bits 7:4 **PUBMDR[3:0]**: PUB moderate revision

Bits 3:0 **PUBMNR[3:0]**: PUB minor revision

14.5.185 DDRPHYC MicroContMuxSel register (DDRPHYC_APBONLY0_MICROCONTMUXSEL)

Address offset: 0x340000

Reset value: 0x0000 0000

PMU config mux select

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MICROCONTMUXSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MICROCONTMUXSEL**:

Controls access to the PHY configuration registers.

1: Microcontroller/PIE has control of csr bus.

0: Microcontroller/PIE csr requests are ignored.

14.5.186 DDRPHYC PMU/controller protocol, controller read-only shadow register (DDRPHYC_APBONLY0_UCTSHADOWREGS)

Address offset: 0x340010

Reset value: 0xXXXX XXXX

This register is used for the mailbox protocol between the firmware and the system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCTDATWRITEPROTSHADOW	UCTWRITEPROTSHADOW
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UCTDATWRITEPROTSHADOW**: Reserved

Bit 0 **UCTWRITEPROTSHADOW**:

0: The PMU has a message for the user.

14.5.187 DDRPHYC DctWriteOnly register (DDRPHYC_APBONLY0_DCTWRITEONLY)

Address offset: 0x3400C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCTWRITEONLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DCTWRITEONLY[15:0]**: Reserved

14.5.188 DDRPHYC DctWriteProt register (DDRPHYC_APBONLY0_DCTWRITEPROT)

Address offset: 0x3400C4

Reset value: 0x0000 0001

DCT downstream mailbox protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCTWRITEPROT
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DCTWRITEPROT**:

By clearing this bit, the user acknowledges the receipt of the message.

14.5.189 DDRPHYC UctWriteOnlyShadow register (DDRPHYC_APBONLY0_UCTWRITEONLYSHADOW)

Address offset: 0x3400C8

Reset value: 0xXXXX XXXX

Read-only view of the UctDatWriteOnly register (reserved for PHY training purpose)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UCTWRITEONLYSHADOW[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **UCTWRITEONLYSHADOW[15:0]**:

Used to pass the message ID for major messages, and to pass the lower 16 bits for streaming messages.

14.5.190 DDRPHYC UctDatWriteOnlyShadow register (DDRPHYC_APBONLY0_UCTDATWRITEONLYSHADOW)

Address offset: 0x3400D0

Reset value: 0xXXXX XXXX

Read-only view of the UctDatWriteOnly register (reserved for PHY training purpose)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UCTDATWRITEONLYSHADOW[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **UCTDATWRITEONLYSHADOW[15:0]**: Used to pass the upper 16 bits for streaming messages. Not used in passing major messages.

14.5.191 DDRPHYC DfiCfgRdDataValidTicks register (DDRPHYC_APBONLY0_DFICFGRDDATAVALIDTICKS)

Address offset: 0x3400DC

Reset value: 0x0000 0006

Number of DfiClk ticks required for valid Rd Data register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFICFGRDDATAVALIDTICKS[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **DFICFGRDDATAVALIDTICKS[5:0]**: Round trip delay of a register read access
This value must not be changed from its reset value.

14.5.192 DDRPHYC reset and clock shutdown control register (DDRPHYC_APBONLY0_MICRORESET)

Address offset: 0x340264

Reset value: 0x0000 0001

Controls reset and clock shutdown on the local microcontroller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESETTOMICRO	RSVDMICRO	TESTWAKEUP	STALLTOMICRO
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RESETTOMICRO**:

This bit is programmable as follows:

- Set this bit to apply synchronous reset to the microcontroller.
- Clear this bit to release reset to the microcontroller.

Note: This reset initializes the program counter to begin execution from the boot vector. This reset also clears the interrupt sticky bits.

Bit 2 **RSVDMICRO**: Reserved

Bit 1 **TESTWAKEUP**: Reserved, must always be set to 0.

Bit 0 **STALLTOMICRO**:

This bit is programmable as follows:

- Set this bit to stall the microcontroller by hardware.
- Clear this bit to allow the microcontroller to continue executing from its current program counter location.

This bit is used at power up to hold the program counter at the boot vector while bios loads the microcontroller program code. While stalled, the microcontroller clocks are gated off for power reduction.

14.5.193 DDRPHYC DfiInitCompleteShadow register (DDRPHYC_APBONLY0_DFIINITCOMPLETESHADOW)

Address offset: 0x3403E8

Reset value: 0xXXXX XXXX

dfi_init_complete, controller read-only shadow

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIINITCOMPLETESHADOW
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DFIINITCOMPLETESHADOW**:

This bit presents a read-only view (a shadow) of DfiInitComplete which is used by the sequencer to control the state of dfi_init_complete.
While MICROCONTMUXSEL = 1, access to this bit does not steal config bus bandwidth. The microcontroller that is polling does not have any performance penalty.

14.5.194 DDRPHYC register map

Table 61. DDRPHYC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x068 + 0x4000*x (x = 0 to 7) Last address: 0x1C068	**DDRPHYC_ANIBx_MTESTMUXSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL[5:0]					
	Reset value																											0	0	0	0	0	0
0x06C-0x098	Reserved	Reserved.																															
0x09C + 0x4000*x (x = 0 to 7) Last address: 0x1C09C	**DDRPHYC_ANIBx_AFORCEDRVCONT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFORCEDRVCONT[3:0]			
	Reset value																													0	0	0	0
0x0A0 + 0x4000*x (x = 0 to 7) Last address: 0x1C0A0	**DDRPHYC_ANIBx_AFORCETRICONT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFORCETRICONT[3:0]			
	Reset value																													0	0	0	0
0x0A4-0x108	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10C + 0x4000*x (x = 0 to 7) Last address: 0x1C10C	**DDRPHYC_ANIBx_ ATXIMPEDANCE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADRVSTRENN [4:0]					ADRVSTRENP [4:0]				
	Reset value																							1	1	1	1	1	1	1	1	1	1
0x110-0x148	Reserved	Reserved.																															
0x14C + 0x4000*x (x = 0 to 7) Last address: 0x1C14C	**DDRPHYC_ANIBx_ ATESTPRBSERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTPRBSERR [3:0]			
	Reset value																													X	X	X	X
0x150	Reserved	Reserved.																															
0x154 + 0x4000*x (x = 0 to 7) Last address: 0x1C154	**DDRPHYC_ANIBx_ ATXSLEWRATE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATXPREDRVMODE [2:0]				ATXPREN[3:0]			ATXPREP[3:0]			
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0x158 + 0x4000*x (x = 0 to 7) Last address: 0x1C158	**DDRPHYC_ANIBx_ ATESTPRBSERR CNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTPRBSERRCNT[15:0]																
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x15C- 0x1FC	Reserved	Reserved.																															
0x200 + 0x4000*x (x = 0 to 7) Last address: 0x1C200	**DDRPHYC_ANIB0_ ATXDLY**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATXDLY[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x204- 0x3FFFC	Reserved	Reserved.																															
0x40000 + 0x4000*x (x = 0 to 1) Last address: 0x44000	**DDRPHYC_DBYTEx _DBYTEMISCMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBYTEDISABLE	Res.	Res.
	Reset value																														0		
0x40004- 0x40064	Reserved	Reserved.																															
0x40068 + 0x4000*x (x = 0 to 1) Last address: 0x44068	**DDRPHYC_DBYTEx _MTESTMUXSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL [5:0]				
	Reset value																																
0x4006C- 0x4007C	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40080 + 0x4000*x (x = 0 to 1) Last address: 0x44080	**DDRPHYC_DBYTEx_DFIMRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIMRL[4:0]				
	Reset value																												0	0	1	1	0
0x40084-0x400FC	Reserved	Reserved.																															
0x40100 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 8) Last address: 0x46100	**DDRPHYC_DBYTEx_VREFDAC0_Ry**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFDAC0[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x40104 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 1) Last address: 0x44504	**DDRPHYC_DBYTEx_TXIMPEDANCECTRL0_By**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRVSTRENDQN[5:0]						DRVSTRENDQP[5:0]					
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x40108	Reserved	Reserved.																															
0x4010C + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 1) Last address: 0x4450C	**DDRPHYC_DBYTEx_DQDQSRCVCNTRL_By**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GAINCURRADJ[4:0]					MAJORMODEDBYTE[2:0]			DFECTRL[1:0]		EXTVREFRANGE	SELANALOGVREF
	Reset value																					0	1	0	1	1	0	1	1	0	0	0	0
0x40110-0x40124	Reserved	Reserved.																															
0x40128 + 0x4000*x (x = 0 to 1) Last address: 0x44128	**DDRPHYC_DBYTEx_DQDQSRCVCNTRL1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENLPREQPDR	RXPADSTANDBYEN	POWERDOWNRCVRDQS	POWERDOWNRCVR[8:0]								
	Reset value																					0	1	0	0	0	0	0	0	0	0	0	0
0x4012C + 0x4000*x (x = 0 to 1) Last address: 0x4412C	**DDRPHYC_DBYTEx_TXIMPEDANCECTRL2_B0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRVSTRENEQLODQN[5:0]						DRVSTRENEQHIDQP[5:0]					
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40130 + 0x4000*x (x = 0 to 1) Last address: 0x4410C	**DDRPHYC_DBYTEx_DQDQSRCVCNTRL2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENRXAGRESSIVEPDR
	Reset value																																0
0x40134 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 1) Last address: 0x44534	**DDRPHYC_DBYTEx_TXODTDRVSTREN_By**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODTSTRENN[5:0]						ODTSTRENP[5:0]					
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x40138-0x40154	Reserved	Reserved.																															
0x40158 + 0x4000*x (x = 0 to 1) Last address: 0x44158	**DDRPHYC_DBYTEx_RXFIFOCHECKSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOLOCUERR	RXFIFOLOCERR
	Reset value																															X	X
0x4015C + 0x4000*x (x = 0 to 1) Last address: 0x4415C	**DDRPHYC_DBYTEx_RXFIFOCHECKERRVALUES**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOWRLOCUERRVALUE[3:0]				RXFIFORDLOCUERRVALUE[3:0]				RXFIFOWRLOCERRVALUE[3:0]				RXFIFORDLOCERRVALUE[3:0]			
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x40160 + 0x4000*x (x = 0 to 1) Last address: 0x44160	**DDRPHYC_DBYTEx_RXFIFOINFO**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOWRLOCU[3:0]				RXFIFORDLOCU[3:0]				RXFIFOWRLOC[3:0]				RXFIFORDLOC[3:0]			
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x40164 + 0x4000*x (x = 0 to 1) Last address: 0x44164	**DDRPHYC_DBYTEx_RXFIFOVISIBILITY**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFORDEN	RXFIFORDPTROVR	RXFIFORDPTR[2:0]		
	Reset value																												0	0	0	0	0
0x40168 + 0x4000*x (x = 0 to 1) Last address: 0x44168	**DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ3210**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOCONTENTSDQ3210[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4016C + 0x4000*x (x = 0 to 1) Last address: 0x4416C	**DDRPHYC_DBYTEx_RXFIFOCONTENTSDQ7654**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOCONTENTSDQ7654[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x40170 + 0x4000*x (x = 0 to 1) Last address: 0x44170	**DDRPHYC_DBYTEx_RXFIFOCONTENTSDBI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFIFOCONTENTSDBI [3:0]			
	Reset value																													-	-	-	-
0x40174-0x40178	Reserved	Reserved.																															
0x4017C + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 1) Last address: 0x4457C	**DDRPHYC_DBYTEx_TXSLEWRATE_By**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPREDRVMODE [2:0]			TXPREN[3:0]				TXPREP[3:0]			
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0x40180-0x4019C	Reserved	Reserved.																															
0x401A0 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 8) Last address: 0x461A0	**DDRPHYC_DBYTEx_RXPBDLYTG0_Ry**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBDLYTG0_R0 [6:0]						
	Reset value																										0	0	0	0	0	0	0
0x401A4 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 8) Last address: 0x461A4	**DDRPHYC_DBYTEx_RXPBDLYTG1_Ry**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBDLYTG1_R0 [6:0]						
	Reset value																										0	0	0	0	0	0	0
0x401A8-0x401FC	Reserved	Reserved.																															
0x40200 + 0x4000*x (x = 0 to 1) Last address: 0x44200	**DDRPHYC_DBYTEx_RXENDLYTG0_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXENDLYTG0_U0[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x40204 + 0x4000*x (x = 0 to 1) Last address: 0x44204	**DDRPHYC_DBYTEx_RXENDLYTG1_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXENDLYTG1_U0[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x40208-0x4022C	Reserved	Reserved.																															
0x40230 + 0x4000*x (x = 0 to 1) Last address: 0x44230	**DDRPHYC_DBYTEx_RXCLKDLYTG0_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG0_U0 [5:0]					
	Reset value																											0	1	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40234 + 0x4000*x (x = 0 to 1) Last address: 0x44234	**DDRPHYC_DBYTEx_RXCLKDLYTG1_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG1_U0[5:0]					
	Reset value																											0	1	0	0	0	0
0x40238-0x4023C	Reserved	Reserved.																															
0x40240 + 0x4000*x (x = 0 to 1) Last address: 0x44240	**DDRPHYC_DBYTEx_RXCLKCDLYTG0_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG0_U0[5:0]					
	Reset value																											0	1	0	0	0	0
0x40244 + 0x4000*x (x = 0 to 1) Last address: 0x44244	**DDRPHYC_DBYTEx_RXCLKCDLYTG1_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG1_U0[5:0]					
	Reset value																											0	1	0	0	0	0
0x40248-0x4027C	Reserved	Reserved.																															
0x40280 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 7) Last address: 0x4429C	**DDRPHYC_DBYTEx_DQyLNSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DQ0LNSEL[2:0]		
	Reset value																														0	0	0
0x40300 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 8) Last address: 0x46300	**DDRPHYC_DBYTEx_TXDQDLYTG0_Ry**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQDLYTG0_R0[8:0]								
	Reset value																								0	0	0	0	1	0	0	0	0
0x40304 + 0x4000*x + 0x400*y (x = 0 to 1; y = 0 to 8) Last address: 0x46304	**DDRPHYC_DBYTEx_TXDQDLYTG1_Ry**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQDLYTG1_R0[8:0]								
	Reset value																								0	0	0	0	1	0	0	0	0
0x40308-0x4033C	Reserved	Reserved.																															
0x40340 + 0x4000*x (x = 0 to 1) Last address: 0x44340	**DDRPHYC_DBYTEx_TXDQSDLYTG0_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG0_U0[9:0]									
	Reset value																							0	0	0	0	0	1	0	0	0	0
0x40344 + 0x4000*x (x = 0 to 1) Last address: 0x44344	**DDRPHYC_DBYTEx_TXDQSDLYTG1_U0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG1_U0[9:0]									
	Reset value																							0	0	0	0	0	1	0	0	0	0
0x40348-0x4038C	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40390 + 0x4000*x (x = 0 to 1) Last address: 0x44390	**DDRPHYC_DBYTEx_DXLCDLSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DXLCDLLIVELOCK	DXLCDLSTICKYUNLOCK	DXLCDLSTICKYLOCK	DXLCDLPHDSNAPVAL	DXLCDLFINESNAPVAL[9:0]									
	Reset value																			X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x40394-0x404FC	Reserved	Reserved.																															
0x40600 + 0x4000*x (x = 0 to 1) Last address: 0x44600	**DDRPHYC_DBYTEx_RXENDLYTG0_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXENDLYTG0_U1[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x40604 + 0x4000*x (x = 0 to 1) Last address: 0x44604	**DDRPHYC_DBYTEx_RXENDLYTG1_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXENDLYTG1_U1[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x40608-0x4062C	Reserved	Reserved.																															
0x40630 + 0x4000*x (x = 0 to 1) Last address: 0x44630	**DDRPHYC_DBYTEx_RXCLKDLYTG0_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG0_U1[5:0]					
	Reset value																											0	1	0	0	0	0
0x40634 + 0x4000*x (x = 0 to 1) Last address: 0x44634	**DDRPHYC_DBYTEx_RXCLKDLYTG1_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKDLYTG1_U1[5:0]					
	Reset value																											0	1	0	0	0	0
0x40638-0x4063C	Reserved	Reserved.																															
0x40640 + 0x4000*x (x = 0 to 1) Last address: 0x44640	**DDRPHYC_DBYTEx_RXCLKCDLYTG0_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG0_U1[5:0]					
	Reset value																											0	1	0	0	0	0
0x40644 + 0x4000*x (x = 0 to 1) Last address: 0x44644	**DDRPHYC_DBYTEx_RXCLKCDLYTG1_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKCDLYTG1_U1[5:0]					
	Reset value																											0	1	0	0	0	0
0x40648-0x4063C	Reserved	Reserved.																															
0x40740 + 0x4000*x (x = 0 to 1) Last address: 0x44740	**DDRPHYC_DBYTEx_TXDQSDLYTG0_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG0_U1[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x40744 + 0x4000*x (x = 0 to 1) Last address: 0x44744	**DDRPHYC_DBYTEx_TXDQSDLYTG1_U1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDQSDLYTG1_U1[10:0]										
	Reset value																						0	0	1	0	0	0	0	0	0	0	0
0x44138-0x7FFFC	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80000	**DDRPHYC_MASTER 0_RXFIFOINIT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INHIBITRXFIFORD	RXFIFOINITPTR
	Reset value																															0	0
0x80004	**DDRPHYC_MASTER 0_FORCECLKDISA BLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCECLKDISABLE [3:0]			
	Reset value																													0	0	0	0
0x80008	Reserved	Reserved.																															
0x8000C	**DDRPHYC_MASTER 0_FORCEINTERNAL UPDATE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCEINTERNALUPDATE
	Reset value																																0
0x80010	**DDRPHYC_MASTER 0_PHYCONFIG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYCONFIGDFI[1:0]		PHYCONFIGDBYTES [3:0]				PHYCONFIGANIBS[3:0]			
	Reset value																							0	1	0	0	1	0	1	0	0	0
0x80014	**DDRPHYC_MASTER 0_PGCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCLKRISEFALLMODE
	Reset value																																0
0x80018	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8001C	**DDRPHYC_MASTER0_TESTBUMPCNTRL1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TESTPOWERGATEEN	TESTEXTVREFRANGE	TESTSELEXTERNALVREF	TESTGAINCURRADJ[4:0]						TESTANALOGOUTCTRL[3:0]				TESTBIASBYPASSEN	TESTMAJORMODE[2:0]		
	Reset value																	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1
0x80020	**DDRPHYC_MASTER0_CALUCLKINFO**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALUCLKTICKSPER1US[10:0]										
	Reset value																						0	1	1	0	0	1	0	0	0	0	0
0x80024	Reserved	Reserved.																															
0x80028	**DDRPHYC_MASTER0_TESTBUMPCNTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCEMTESTONALERT	TESTBUMPDATASEL[5:0]						TESTBUMPTOGGLE	TESTBUMPEN[1:0]	
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x8002C	**DDRPHYC_MASTER0_SEQ0BDLY0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ0BDLY0[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80030	**DDRPHYC_MASTER0_SEQ0BDLY1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ0BDLY1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80034	**DDRPHYC_MASTER0_SEQ0BDLY2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ0BDLY2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80038	**DDRPHYC_MASTER0_SEQ0BDLY3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ0BDLY3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8003C	**DDRPHYC_MASTER0_PHYALERTSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYALERT
	Reset value																																x
0x80040	**DDRPHYC_MASTER0_PPTTRAINSETUP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYMSTRMAXREQTOACK[2:0]			PHYMSTRTRAININTERVAL[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x80044	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80048	**DDRPHYC_MASTER0_ATESTMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTMODESEL[2:0]			ATESTCLKEN	ATESTPRBSEN
	Reset value																												0	0	0	0	0
0x8004C	Reserved	Reserved.																															
0x80050	**DDRPHYC_MASTER0_TXCALBINP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINP[4:0]				
	Reset value																												X	X	X	X	X
0x80054	**DDRPHYC_MASTER0_TXCALBINN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINN[4:0]				
	Reset value																												X	X	X	X	X
0x80058	**DDRPHYC_MASTER0_TXCALPOVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINPOVREN	TXCALBINPOVRVAL[4:0]				
	Reset value																											0	0	0	0	0	0
0x8005C	**DDRPHYC_MASTER0_TXCALNOVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCALBINNOVREN	TXCALBINNOVRVAL[4:0]				
	Reset value																											0	0	0	0	0	0
0x80060	**DDRPHYC_MASTER0_DFIMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI1OVERRIDE	DFI1ENABLE	DFI0ENABLE
	Reset value																														0	0	1
0x80064	**DDRPHYC_MASTER0_TRISTATEMODECA**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKDISVAL[1:0]		DDR2TMODE	DISDYNADRTRI
	Reset value																													0	0	0	0
0x80068	**DDRPHYC_MASTER0_MTESTMUXSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTMUXSEL[5:0]					
	Reset value																											0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8006C	**DDRPHYC_MASTER0_MTESTPGMINFO**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTPGMINFO
	Reset value																																0
0x80070	**DDRPHYC_MASTER0_DYNPWRDNUP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DYNPOWERDOWN
	Reset value																																0
0x80074	**DDRPHYC_MASTER0_PMIENABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMIENABLE
	Reset value																																0
0x80078	**DDRPHYC_MASTER0_PHYTID**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYTID[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8007C	Reserved	Reserved.																															
0x80080	**DDRPHYC_MASTER0_HWTMRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTMRL[4:0]				
	Reset value																												0	0	1	1	0
0x80084	**DDRPHYC_MASTER0_DFIPHYUPD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIPHYUPDINTTHRESHOLD[3:0]				DFIPHYUPDTHRESHOLD[3:0]				DFIPHYUPDMODE	DFIPHYUPDRESP[2:0]			DFIPHYUPDCNT[3:0]			
	Reset value																	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1
0x80088	**DDRPHYC_MASTER0_PDAMRSWRITEMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDAMRSWRITEMODE
	Reset value																																0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8008C	**DDRPHYC_MASTER 0_DFIGEARDOWN CTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIGEARDOWNCTL [1:0]	
	Reset value																															0	0
0x80090	**DDRPHYC_MASTER 0_DQSPREAMBLE CONTROL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDQSEXTENSION	LP4STTCPREBRIDGERXEN	LP4POSTAMBLEEXT	LP4TGLTWOTCKTXDQSPRE	POSITIONDFEINIT[2:0]		TWOTCKTXDQSPRE	TWOTCKRXDQSPRE
	Reset value																								0	0	0	0	0	1	0	0	0
0x80094	**DDRPHYC_MASTER 0_MASTERX4 CONFIG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X4TG[3:0]			
	Reset value																													0	0	0	0
0x80098	**DDRPHYC_MASTER 0_WRLEVBITS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRLEVFORDQSU [3:0]				WRLEVFORDQSL [3:0]			
	Reset value																									1	0	0	1	1	0	0	0
0x8009C	**DDRPHYC_MASTER 0_ENABLECS MULTICAST**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLECSMULTICAST
	Reset value																																0
0x800A0-0x800AC	Reserved	Reserved.																															
0x800B0	**DDRPHYC_MASTER 0_ACX4ANIBDIS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACX4ANIBDIS[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x800B4	**DDRPHYC_MASTER 0_DMIPINPRESENT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDDBIENABLED
	Reset value																																0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x800B8	**DDRPHYC_MASTER0_ARDPTRINITVAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARDPTRINITVAL[3:0]			
	Reset value																													0	0	0	0
0x800BC-0x800E4	Reserved	Reserved.																															
0x800E8	**DDRPHYC_MASTER0_DBYTEDLLMODECNTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		DLLRXPREAMBLEMODE	Res.
	Reset value																															0	
0x800EC-0x80110	Reserved	Reserved.																															
0x80114	**DDRPHYC_MASTER0_CALOFFSETS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALDRVPUTHOFFSET[3:0]				CALDRVPDTHOFFSET[3:0]				CALCMPR5OFFSET[5:0]						
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80118	Reserved	Reserved.																															
0x8011C	**DDRPHYC_MASTER0_SARINITVALS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SARINITPEXT[2:0]			SARINITNINT[2:0]			SARINITOFFSET05[2:0]		
	Reset value																								1	0	0	1	0	0	1	1	1
0x80120	Reserved	Reserved.																															
0x80124	**DDRPHYC_MASTER0_CALPEXTOVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALPEXTOVR[4:0]				
	Reset value																												0	0	0	0	0
0x80128	**DDRPHYC_MASTER0_CALCMPR5OVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALCMPR5OVR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x8012C	**DDRPHYC_MASTER0_CALNINTOVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALNINTOVR[4:0]				
	Reset value																												0	0	0	0	0
0x80130-0x8013C	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80140	**DDRPHYC_MASTER 0_CALDRVSTR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALDRVSTRPU50 [3:0]				CALDRVSTRPD50 [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x80144-0x80154	Reserved	Reserved.																															
0x80158	**DDRPHYC_MASTER 0_PROCODTTIME CTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PODTTAILWIDTHEXT [1:0]		PODTSTARTDELAY [1:0]		PODTTAILWIDTH [1:0]	
	Reset value																											0	1	1	1	1	1
0x8015C-0x80168	Reserved	Reserved.																															
0x8016C	**DDRPHYC_MASTER 0_MEMALERT CONTROL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MALERTFORCEERROR	MALERTDISABLEVAL	MALERTRXEN	MALERTPUEN	MALERTPUSTREN [3:0]				MALERTVREFEXTEN	MALERTVREFLEVEL [6:0]						
	Reset value																	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80170	**DDRPHYC_MASTER 0_MEMALERT CONTROL2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MALERTSYNCBYPASS
	Reset value																																0
0x80174-0x8017C	Reserved	Reserved.																															
0x80180	**DDRPHYC_MASTER 0_MEMRESETL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTECTMEMRESET	MEMRESETLVALUE
	Reset value																															0	0
0x80184-0x801B4	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x801B8	**DDRPHYC_MASTER0_PUBMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTMEMSRC
	Reset value																																0
0x801BC	**DDRPHYC_MASTER0_MISCPHYSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORMEMRESET	DCTSANE
	Reset value																															X	X
0x801C0	**DDRPHYC_MASTER0_CORELOOPBACKSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CORELOOPBACKSEL
	Reset value																																0
0x801C4	**DDRPHYC_MASTER0_DLLTRAINPARAM**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTENDPHDTIME[1:0]	
	Reset value																															1	0
0x801C8-0x801CC	Reserved	Reserved.																															
0x801D0	**DDRPHYC_MASTER0_HWTLPCSENBYPASS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTLPCSENBYPASS
	Reset value																																0
0x801D4	**DDRPHYC_MASTER0_DFICAMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFID4ALTCAMODE	DFILP4CAMODE	DFID4CAMODE	DFILP3CAMODE
	Reset value																													0	0	0	0
0x801D8-0x801DC	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x801E0	**DDRPHYC_MASTER 0_DLLCONTROL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLLRESETRSVD	DLLRESETSLAVE	DLLRESETRELOCK
	Reset value																														0	1	1
0x801E4	**DDRPHYC_MASTER 0_PULSEDLL UPDATEPHASE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALWAYSUPDATELCDLPHASE	TRAINUPDATEPHASEONLONGBUBBLE	UPDATEPHASEDESTRESERVED[2:0]			PULSEACADLLUPDATEPHASE	PULSEACKDLLUPDATEPHASE	PULSEDBYTEDLLUPDATEPHASE
	Reset value																									0	0	0	0	0	0	0	0
0x801E8-0x801EC	Reserved	Reserved.																															
0x801F0	**DDRPHYC_MASTER 0_DLLGAINCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLLSEEDSEL [3:0]				DLLGAINTV [3:0]				DLLGAINIV [3:0]			
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	1
0x801F4-0x8021C	Reserved	Reserved.																															
0x80220	**DDRPHYC_MASTER 0_CALRATE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLEBACKGROUNDZQUPDATES	CALONCE	CALRUN	CALINTERVAL[3:0]			
	Reset value																										0	0	0	1	0	0	1
0x80224	**DDRPHYC_MASTER 0_CALZAP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALZAP
	Reset value																																0
0x80228	Reserved	Reserved.																															
0x8022C	**DDRPHYC_MASTER 0_PSTATE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTATE [3:0]			
	Reset value																													1	1	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80230	Reserved	Reserved.																															
0x80234	**DDRPHYC_MASTER 0_PLLOUTGATE CONTROL**	Res.																														RESERVED2X1	PCLKGATEEN
	Reset value																															0	0
0x80238-0x8023C	Reserved	Reserved.																															
0x80240	**DDRPHYC_MASTER 0_PORCONTROL**	Res.																															PLLDLLLOCKDONE
	Reset value																																0
0x80244-0x80258	Reserved	Reserved.																															
0x8025C	**DDRPHYC_MASTER 0_CALBUSY**	Res.																															CALBUSY
	Reset value																																X
0x80260	**DDRPHYC_MASTER 0_CALMISC2**	Res.																RESERVED15	CALSLOWCMPANA	CALCANCELROUNDERRDIS	CALCMPTRRESTRIM	RESERVED11	RESERVED10X3[7:0]								CALNUMVOTES[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x80264	Reserved	Reserved.																															
0x80268	**DDRPHYC_MASTER 0_CALMISC**	Res.																													CALPEXTDIS	CALNINTDIS	CALCMPR5DIS
	Reset value																														0	0	0
0x8026C	**DDRPHYC_MASTER 0_CALVREFS**	Res.																														CALVREFS [1:0]	
	Reset value																															1	0
0x80270	**DDRPHYC_MASTER 0_CALCMPR5**	Res.																								CALCMPR5[7:0]							
	Reset value																									X	X	X	X	X	X	X	X

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80274	**DDRPHYC_MASTER 0_CALNINT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALNINTTHB [4:0]				
	Reset value																												X	X	X	X	X
0x80278	**DDRPHYC_MASTER 0_CALPEXT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALPEXTTHB [4:0]				
	Reset value																												X	X	X	X	X
0x8027C-0x8029C	Reserved	Reserved.																															
0x802A0	**DDRPHYC_MASTER 0_CALCMPINVERT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMPINVERTCALODTPU	CMPINVERTCALODTPD	CMPINVERTCALDRVPU50	CMPINVERTCALDRVPD50	CMPINVERTCALDAC50
	Reset value																												0	0	0	1	1
0x802A4-0x802B4	Reserved	Reserved.																															
0x802B8	**DDRPHYC_MASTER 0_CALCMPANA CNTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMPRBIASBYPASSEN	CMPRGAINRESADJ	CMPRGAINCURRADJ [7:0]							
	Reset value																							0	1	0	0	0	1	0	0	1	0
0x802BC	Reserved	Reserved.																															
0x802C0	**DDRPHYC_MASTER 0_DFIRDDATACSDE STMAP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIRDDESTM3 [1:0]		DFIRDDESTM2 [1:0]		DFIRDDESTM1 [1:0]		DFIRDDESTM0 [1:0]	
	Reset value																									1	1	1	0	0	1	0	0
0x802C4	Reserved	Reserved.																															
0x802C8	**DDRPHYC_MASTER 0_VREFINGLOBAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOBALVREFINMODE	GLOBALVREFINTRIM [3:0]				GLOBALVREFINDAC [6:0]							GLOBALVREFINSEL [2:0]		
	Reset value																		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x802CC	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x802D0	**DDRPHYC_MASTER0_DFIWRDATACSDESTMAP**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIWRDESTM3[1:0]		DFIWRDESTM2[1:0]		DFIWRDESTM1[1:0]		DFIWRDESTM0[1:0]	
	Reset value																									1	1	1	0	0	1	0	0
0x802D4	**DDRPHYC_MASTER0_MASUPDGOODCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASUPDGOODCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802D8	**DDRPHYC_MASTER0_PHYUPD0GOODCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPD0GOODCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802DC	**DDRPHYC_MASTER0_PHYUPD1GOODCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPD1GOODCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802E0	**DDRPHYC_MASTER0_CTLUPD0GOODCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTLUPD0GOODCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802E4	Reserved	Reserved.																															
0x802E8	**DDRPHYC_MASTER0_MASUPDFAILCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASUPDFAILCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802EC	**DDRPHYC_MASTER0_PHYUPD0FAILCTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPD0FAILCTR[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x802F0	Reserved	Reserved.																															
0x802F4	**DDRPHYC_MASTER0_PHYPERFCTRENABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPD1FAILCTL	PHYUPD0FAILCTL	MASUPDFAILCTL	CTLUPD1GOODCTL	CTLUPD0GOODCTL	PHYUPD1GOODCTL	PHYUPD0GOODCTL	MASUPDGOODCTL
	Reset value																									0	0	0	0	0	0	0	0
0x802F8-0x80308	Reserved	Reserved.																															
0x8030C	**DDRPHYC_MASTER0_PLLPWRDN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLPWRDN
	Reset value																																1

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80310	**DDRPHYC_MASTER 0_PLLRESET**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLRESET
	Reset value																																1
0x80314	**DDRPHYC_MASTER 0_PLLCTRL2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLFREQSEL[4:0]				
	Reset value																												1	1	0	0	1
0x80318	**DDRPHYC_MASTER 0_PLLCTRL0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLSPARECTRL0	PLLLOCKPHSEL[1:0]		PLLLOCKCNTSEL	PLLGEARSHIFT	PLLRESERVED10X9[1:0]		PLLSYNCBUSBYP	PLLSYNCBUSFLUSH	PLLSELDFIFREQRATIO	PLLBYPASSMODE	PLLPRESET	PLLOUTBYPEN	PLLX2MODE	PLLBYPSEL	PLLSTANDBY
	Reset value																	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
0x8031C	**DDRPHYC_MASTER 0_PLLCTRL1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLCPPROPCTRL[3:0]				PLLCPINTCTRL[4:0]				
	Reset value																								0	0	0	1	0	0	0	0	0
0x80320	**DDRPHYC_MASTER 0_PLLTST**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLDIGTSTSEL[3:0]				PLLANATSTSEL[3:0]				PLLANATSTEN
	Reset value																								0	0	0	0	0	0	0	0	0
0x80324	**DDRPHYC_MASTER 0_PLLLOCKSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLLOCKSTATUS
	Reset value																																X
0x80328	**DDRPHYC_MASTER 0_PLLTESTMODE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLTESTMODE[15:0]															
	Reset value																	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8032C	**DDRPHYC_MASTER 0_PLLCTRL3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLENCAL	PLLFORCECAL	PLLDACVALIN [4:0]					PLLMAXRANGE [4:0]					PLLSPARE[3:0]			
	Reset value																	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0x80330	**DDRPHYC_MASTER 0_PLLCTRL4**	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PLLCPPROPGSCTRL [3:0]				PLLCPINTGSCTRL [4:0]				
	Reset value																								1	0	1	1	1	1	1	1	1
0x80334	**DDRPHYC_MASTER 0_PLLENDOFCAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLENDOFCAL
	Reset value																																X
0x80338	**DDRPHYC_MASTER 0_PLLSTANDBYEFF**	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PLLSTANDBYEFF
	Reset value																																X
0x8033C	**DDRPHYC_MASTER 0_PLLDACVALOUT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLDACVALOUT[4:0]				
	Reset value																												-	-	-	-	-
0x80340-0x80388	Reserved	Reserved.																															
0x8038C	**DDRPHYC_MASTER 0_LCDLDBGCNTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCDLSTATUSSEL [3:0]				LCDLTSTENABLE	LCDLFINESNAP	LCDLFINEOVR	LCDLFINEOVRVAL[8:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80390	**DDRPHYC_MASTER 0_ACLCDLSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLCDLLIVELOCK	ACLCDLSTICKYUNLOCK	ACLCDLSTICKYLOCK	ACLCDLPHDSNAPVAL	ACLCDLFINESNAPVAL[9:0]									
	Reset value																			X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x80394-0x803B0	Reserved	Reserved.																															
0x803B4	**DDRPHYC_MASTER 0_CUSTPHYREV**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUSTPHYREV [5:0]					
	Reset value																											X	X	X	X	X	X
0x803B8	**DDRPHYC_MASTER 0_PHYREV**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYMJR[7:0]								PHYMDR [3:0]				PHYMNR [3:0]			
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x803BC	**DDRPHYC_MASTER 0_LP3EXITSEQ0B STARTVECTOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LP3EXITSEQ0BSTARTVECPLLBYPASSED [3:0]				LP3EXITSEQ0BSTARTVECPLLENABLED [3:0]			
	Reset value																									1	1	0	1	1	1	1	0
0x803C0	**DDRPHYC_MASTER 0_DFIFREQXLAT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL3 [3:0]				DFIFREQXLATVAL2 [3:0]				DFIFREQXLATVAL1 [3:0]				DFIFREQXLATVAL0 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803C4	**DDRPHYC_MASTER 0_DFIFREQXLAT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL7 [3:0]				DFIFREQXLATVAL6 [3:0]				DFIFREQXLATVAL5 [3:0]				DFIFREQXLATVAL4 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x803C8	**DDRPHYC_MASTER 0_DFIFREQXLAT2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL11 [3:0]				DFIFREQXLATVAL10 [3:0]				DFIFREQXLATVAL9 [3:0]				DFIFREQXLATVAL8 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803CC	**DDRPHYC_MASTER 0_DFIFREQXLAT3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL15 [3:0]				DFIFREQXLATVAL14 [3:0]				DFIFREQXLATVAL13 [3:0]				DFIFREQXLATVAL12 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803D0	**DDRPHYC_MASTER 0_DFIFREQXLAT4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL19 [3:0]				DFIFREQXLATVAL18 [3:0]				DFIFREQXLATVAL17 [3:0]				DFIFREQXLATVAL16 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803D4	**DDRPHYC_MASTER 0_DFIFREQXLAT5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL23 [3:0]				DFIFREQXLATVAL22 [3:0]				DFIFREQXLATVAL21 [3:0]				DFIFREQXLATVAL20 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803D8	**DDRPHYC_MASTER 0_DFIFREQXLAT6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL27 [3:0]				DFIFREQXLATVAL26 [3:0]				DFIFREQXLATVAL25 [3:0]				DFIFREQXLATVAL24 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x803DC	**DDRPHYC_MASTER 0_DFIFREQXLAT7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQXLATVAL31 [3:0]				DFIFREQXLATVAL30 [3:0]				DFIFREQXLATVAL29 [3:0]				DFIFREQXLATVAL28 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x803E0	**DDRPHYC_MASTER0_TXRDPTRINIT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXRDPTRINIT
	Reset value																																1
0x803E4	**DDRPHYC_MASTER0_DFIINIT COMPLETE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIINITCOMPLETE
	Reset value																																0
0x803E8	**DDRPHYC_MASTER0_DFIFREQRATIO**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIFREQRATIO [1:0]	
	Reset value																														0	1	
0x803EC	**DDRPHYC_MASTER0_RXFIFOCHECKS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOFREQUENTRXFIFOCHECKS
	Reset value																																0
0x803F0-0x803F8	Reserved	Reserved.																															
0x803FC	**DDRPHYC_MASTER0_MTESTDTOCTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MTESTDTOEN
	Reset value																																0
0x80400	**DDRPHYC_MASTER0_MAPCAA0TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA0TODFI [3:0]			
	Reset value																													0	0	0	0
0x80404	**DDRPHYC_MASTER0_MAPCAA1TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA1TODFI [3:0]			
	Reset value																													0	0	0	1

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80408	**DDRPHYC_MASTER 0_MAPCAA2TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA2TODFI [3:0]			
	Reset value																													0	0	1	0
0x8040C	**DDRPHYC_MASTER 0_MAPCAA3TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA3TODFI [3:0]			
	Reset value																													0	0	1	1
0x80410	**DDRPHYC_MASTER 0_MAPCAA4TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA4TODFI [3:0]			
	Reset value																													0	1	0	0
0x80414	**DDRPHYC_MASTER 0_MAPCAA5TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA5TODFI [3:0]			
	Reset value																													0	1	0	1
0x80418	**DDRPHYC_MASTER 0_MAPCAA6TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA6TODFI [3:0]			
	Reset value																													0	1	1	0
0x8041C	**DDRPHYC_MASTER 0_MAPCAA7TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAA7TODFI [3:0]			
	Reset value																													0	1	1	1
0x80420-0x80440	Reserved	Reserved.																															
0x80444	**DDRPHYC_MASTER 0_MAPCAB1TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB1TODFI [3:0]			
	Reset value																													0	0	0	1

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80448	**DDRPHYC_MASTER0_MAPCAB2TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB2TODFI [3:0]			
	Reset value																													0	0	1	0
0x8044C	**DDRPHYC_MASTER0_MAPCAB3TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB3TODFI [3:0]			
	Reset value																													0	0	1	1
0x80450	**DDRPHYC_MASTER0_MAPCAB4TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB4TODFI [3:0]			
	Reset value																													0	1	0	0
0x80454	**DDRPHYC_MASTER0_MAPCAB5TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB5TODFI [3:0]			
	Reset value																													0	1	0	1
0x80458	**DDRPHYC_MASTER0_MAPCAB6TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB6TODFI [3:0]			
	Reset value																													0	1	1	0
0x8045C	**DDRPHYC_MASTER0_MAPCAB7TODFI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAPCAB7TODFI [3:0]			
	Reset value																													0	1	1	1
0x80460-0x80468	Reserved	Reserved.																															
0x8046C	**DDRPHYC_MASTER0_PHYINTERRUPTENABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYHWRESERVEDEN [4:0]					PHYRXFIFOCHECKEN	PHYVTDRIFTALARMEN [1:0]		PHYFWRESERVEDEN [4:0]					PHYTRNGFAILEN	PHYINITCMPLTEN	PHYTRNGCMPLTEN
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80470	**DDRPHYC_MASTER0_PHYINTERRUPTFWCONTROL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYFWRESERVEDFW[4:0]					PHYTRNGFAILFW	PHYINITCMPLTFW	PHYTRNGCMPLTFW
	Reset value																									0	0	0	0	0	0	0	0
0x80474	**DDRPHYC_MASTER0_PHYINTERRUPTMASK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYHWRESERVEDMSK[4:0]					PHYRXFIFOCHECKMSK	PHYVTDRIFTALARMMSK[1:0]		PHYFWRESERVEDMSK[4:0]					PHYTRNGFAILMSK	PHYINITCMPLTMSK	PHYTRNGCMPLTMSK
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80478	**DDRPHYC_MASTER0_PHYINTERRUPTCLEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYHWRESERVEDCLR[4:0]					PHYRXFIFOCHECKCLR	PHYVTDRIFTALARMCLR[1:0]		PHYFWRESERVEDCLR[4:0]					PHYTRNGFAILCLR	PHYINITCMPLTCLR	PHYTRNGCMPLTCLR
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8047C	**DDRPHYC_MASTER0_PHYINTERRUPTSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYHWRESERVED[4:0]					PHYRXFIFOCHECK	VTDRIFTALARM[1:0]		PHYFWRESERVED[4:0]					PHYTRNGFAIL	PHYINITCMPLT	PHYTRNGCMPLT
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x80480	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS0[4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80484	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS1 [4:0]				
	Reset value																												0	0	0	0	0
0x80488	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS2 [4:0]				
	Reset value																												0	0	0	0	0
0x8048C	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS3 [4:0]				
	Reset value																												0	0	0	0	0
0x80490	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS4 [4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80494	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS5[4:0]				
	Reset value																												0	0	0	0	0
0x80498	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS6[4:0]				
	Reset value																												0	0	0	0	0
0x8049C	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS7[4:0]				
	Reset value																												0	0	0	0	0
0x804A0	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS8**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS8[4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804A4	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS9**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS9[4:0]				
	Reset value																												0	0	0	0	0
0x804A8	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS10**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS10[4:0]				
	Reset value																												0	0	0	0	0
0x804AC	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS11[4:0]				
	Reset value																												0	0	0	0	0
0x804B0	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS12**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS12[4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804B4	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS13**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS13[4:0]				
	Reset value																												0	0	0	0	0
0x804B8	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS14**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS14[4:0]				
	Reset value																												0	0	0	0	0
0x804BC	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS15**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS15[4:0]				
	Reset value																												0	0	0	0	0
0x804C0	**DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS17**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTADDRESS17[4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804C4	**DDRPHYC_MASTER0_HWTSWIZZLEHWTACTN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTACTN[4:0]				
	Reset value																												0	0	0	0	0
0x804C8	**DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK0[4:0]				
	Reset value																												0	0	0	0	0
0x804CC	**DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK1[4:0]				
	Reset value																												0	0	0	0	0
0x804D0	**DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBANK2[4:0]				
	Reset value																												0	0	0	0	0
0x804D4	**DDRPHYC_MASTER0_HWTSWIZZLEHWTBG0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBG0[4:0]				
	Reset value																												0	0	0	0	0

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804D8	**DDRPHYC_MASTER0_HWTSWIZZLEHWTBG1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTBG1[4:0]				
	Reset value																												0	0	0	0	0
0x804DC	**DDRPHYC_MASTER0_HWTSWIZZLEHWTCASN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTCASN[4:0]				
	Reset value																												0	0	0	0	0
0x804E0	**DDRPHYC_MASTER0_HWTSWIZZLEHWTRASN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTRASN[4:0]				
	Reset value																												0	0	0	0	0
0x804E4	**DDRPHYC_MASTER0_HWTSWIZZLEHWTWEN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTWEN[4:0]				
	Reset value																												0	0	0	0	0
0x804E8	**DDRPHYC_MASTER0_HWTSWIZZLEHWTPARITYIN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HWTSWIZZLEHWTPARITYIN[4:0]				
	Reset value																												0	0	0	0	0
0x804EC	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804F0	**DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPDREQDELAY0 [3:0]				CTRLUPDACKDELAY0 [3:0]				PHYUPDREQDELAY0 [3:0]				PHYUPDACKDELAY0 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x804F4	**DDRPHYC_MASTER0_DFIHANDSHAKEDELAYS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYUPDREQDELAY1 [3:0]				PHYUPDACKDELAY1 [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x804F8	**DDRPHYC_MASTER0_REMOTEIMPCAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLAVECODEUPDATED	CALIBSLAVE
	Reset value																															X	X
0x804FC-0x24009C	Reserved	Reserved.																															
0x2400A0	**DDRPHYC_INITENG0_PHYINLP3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PHYINLP3
	Reset value																																1
0x2400A4-0x3001FC	Reserved	Reserved.																															
0x300200	**DDRPHYC_DRTUB0_UCCLKHCLKENABLES**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HCLKEN	UCCLKEN
	Reset value																															1	1
0x300204	**DDRPHYC_DRTUB0_CURPSTATE0B**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURPSTATE0B [3:0]			
	Reset value																													0	0	0	0
0x300208-0x3003B0	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3003B4	**DDRPHYC_DRTUB0_CUSTPUBREV**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUSTPUBREV [5:0]					
	Reset value																											X	X	X	X	X	X
0x3003B8	**DDRPHYC_DRTUB0_PUBREV**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUBMJR[7:0]								PUBMDR [3:0]				PUBMNR [3:0]			
	Reset value																	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
0x3003BC-0x33FFFC	Reserved	Reserved.																															
0x340000	**DDRPHYC_APBONLY0_MICROCONTMUXSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MICROCONTMUXSEL
	Reset value																																0
0x340004-0x34000C	Reserved	Reserved.																															
0x340010	**DDRPHYC_APBONLY0_UCTSHADOWREGS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCTDATWRITEPROTSHADOW	UCTWRITEPROTSHADOW
	Reset value																															X	X
0x340014-0x3400BC	Reserved	Reserved.																															
0x3400C0	**DDRPHYC_APBONLY0_DCTWRITEONLY**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCTWRITEONLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3400C4	**DDRPHYC_APBONLY0_DCTWRITEPROT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCTWRITEPROT
	Reset value																																1
0x3400C8	**DDRPHYC_APBONLY0_UCTWRITEONLYSHADOW**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCTWRITEONLYSHADOW[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3400CC	Reserved	Reserved.																															

Table 61. DDRPHYC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3400D0	**DDRPHYC_APBONLY0_UCTDATWRITEONLYSHADOW**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCTDATWRITEONLYSHADOW[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3400D4-0x3400D8	Reserved	Reserved.																															
0x3400DC	**DDRPHYC_APBONLY0_DFICFGRDDATAVALIDTICKS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFICFGRDDATAVALIDTICKS[5:0]					
	Reset value																											0	0	0	1	1	0
0x3400E0-0x340260	Reserved	Reserved.																															
0x340264	**DDRPHYC_APBONLY0_MICRORESET**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESETTOMICRO	RSVDMICRO	TESTWAKEUP	STALLTOMICRO
	Reset value																													0	0	0	1
0x340268-0x3403E4	Reserved	Reserved.																															
0x3403E8	**DDRPHYC_APBONLY0_DFIINITCOMPLETESHADOW**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFIINITCOMPLETESHADOW
	Reset value																																x

Refer to *Section 2.4* for the register boundary addresses.

15 DDR performance monitor (DDRPERFM)

15.1 DDRPERFM introduction

The DDRPERFM is intended to monitor DDR controller performance by counting events from a set of signals and gathering results under the software control. The DDRPERFM can be used to feedback important statistics such as the DDR utilization, read/write bandwidth, bank activation, idle state duration for some off-line analysis of the DDR system performance and for optimal parameter settings to improve the DDR utilization, and for power optimization.

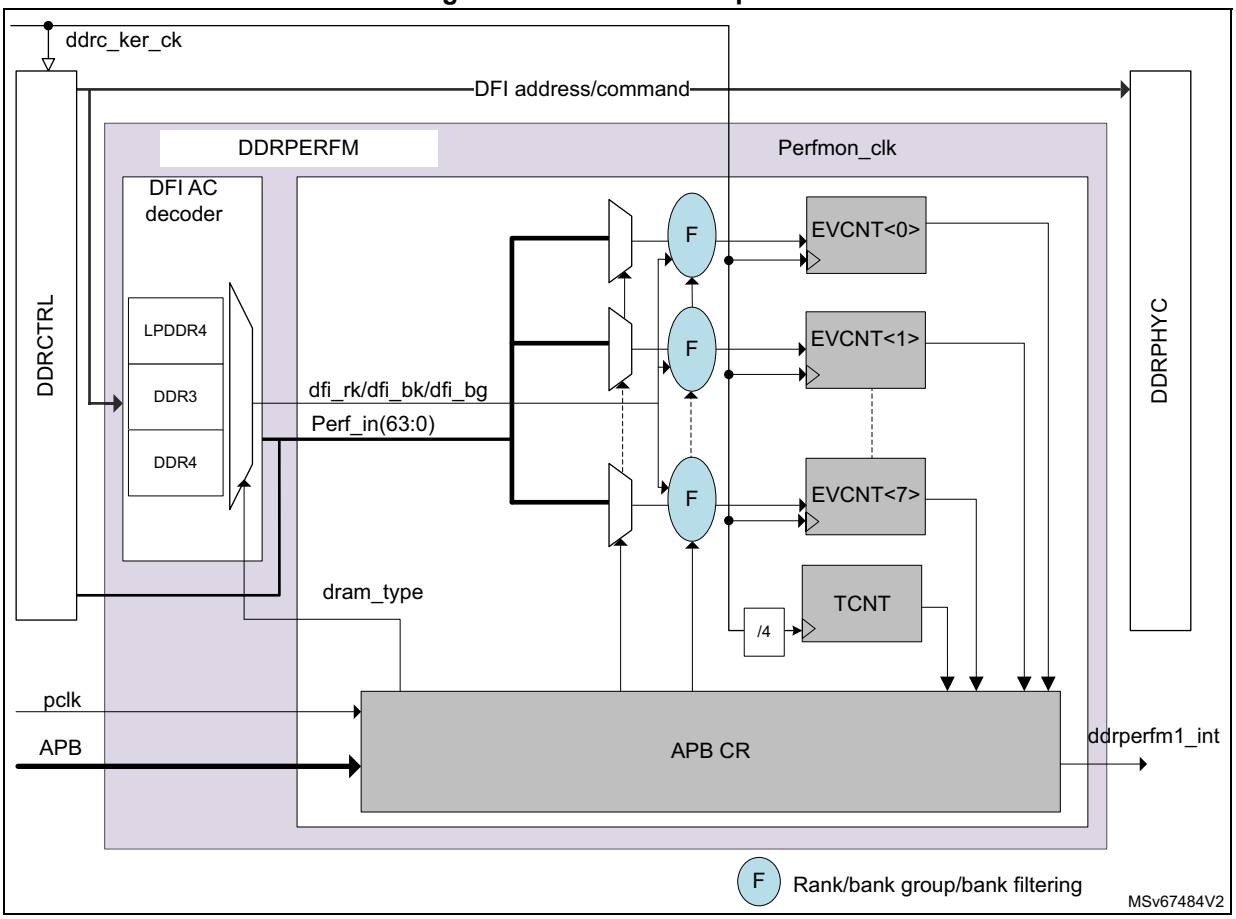
15.2 DDRPERFM main features

- 32-bit time counter (TCNT) clocked at DFI_CLK/4 (that is ddrc_ker_clk/4)
- Eight 32-bit event counters (EVCNT0 to EVCNT7) clocked at DFI_CLK.
- For each of the 8 counters, selection among 64 signals:
  - Either selection of performance logging signals exported by DDRCTRL
  - Information generated decoding DFI commands between DDRCTRL and DDRPHYC
  - Counters that can be individually enabled
- Programmable rank, bank group (only for DDR4), and bank address filtering module in front of each event counter
  - The filter only applies to commands monitored at the DFI interface.
- Global start/stop triggered by software
- Automatic stop when the time counter reaches a programmable value or when an event counter overflows
- All counters are frozen and a dedicated (maskable) interruption line is asserted when a stop condition occurs.
- APB interface: 1 Kbyte used for control register mapping
  - The performance monitor interface is shared with DDRCTRL and the mode register read registers. The performance monitor base address is 0x1000 (above DDRCTRL registers), then the address space is from 0x1000 to 0x13FF.

15.3 DDRPERFM block diagram

Figure 44 presents the overall architecture of the DDRPERFM. The DDRPERFM either sniffs signals on the DFI interface, or receives inputs from the DDRCTRL performance port. Based on the filtering scheme programmed in its registers, the DDRPERFM counts filtered event occurrences in one of its eight internal counters. An interrupt is fired when a stop condition occurs (the counter overflows or the stop is reached).

Figure 44. DDRPERFM top view



15.4 DDRPERFM functional description

A free running 32-bit time counter (TCNT) shares the DDRCTRL counter, and enables to compute the DDRSS statistics. The maximum value is programmable through the APB register. The default value is 0xFFFF FFFF. The TCNT is incremented every 4 DFI_CLK (= 8 DRAM_CLK cycles = burst length). When the TCNT overflows, all counters are frozen. The counter values can be read through the APB interface.

The DDRPERM implements eight independent 32-bit counters: EVCNT<7-0> sharing DDRCTRL clock. The counters input selection is configured through a dedicated control register (SEL_EVENT<I>) which enables to choose one of the 64 events that are available.

For each counter, a filtering can be applied based on bank or bank group (for DDR4 only). This filtering is managed through FILT_EVENT<i>.

The EVTCNT counters can be cleared by software through the APB interface, and their values can be read through the APB registers.

The DDRPERFM is inactive by default. It starts counting events when a start triggered by software is received. Counters are stopped when a stop is triggered by the software.

When the counter overflows, all counters are frozen and a dedicated maskable interruption line is asserted (ddrperfm_int). When an interrupt is detected, before clearing the interrupt, the software must clear the counters to avoid the interrupt to be raised again.

15.4.1 DDRPERFM performance monitor events

The commands sent by the controller can be monitored either at the DFI interface (signal name DFI_xxx), or at the performance logging interface (signal name OP_xxx).

A better granularity is provided for commands captured at DFI interface:

- ability to discriminate between read/write commands with and without auto-precharge
- ability to filter activate, precharge, read, write, and refresh commands per bank

The list of events which can be captured by the performance monitor is given in the tables below:

- *Table 62* lists the events generated from the DFI command address decoder.
- *Table 63* lists the events generated natively by the DDRCTRL (with event number).

Table 62. Signals coming from DFI command-address decoder

-	N^0	Signal name	Description
DFI decoded DRAM commands	0	DFI_IS_ACT	Asserted when an ACT command is sent by the controller
	1	DFI_IS_PREPB	Asserted when a PRE per bank command is sent by the controller
	2	DFI_IS_PREAB	Asserted when a PRE all banks command is sent by the controller
	3	DFI_IS_RD	Asserted when a RD command is sent by the controller (AP bit ignored)
	4	DFI_IS_RDA	Asserted when a RD with AP command is sent by the controller
	5	Reserved	Reserved
	6	DFI_IS_WR	Asserted when a WR command is sent by the controller (AP bit ignored)
	7	DFI_IS_WRA	Asserted when a WR with AP command is sent by the controller
	8	Reserved	Reserved
	9	DFI_IS_MWR	Asserted when a MWR command is sent by the controller (AP bit ignored, LPDDR4 only)
	10	DFI_IS_MWRA	Asserted when a MWR with AP command is sent by the controller (LPDDR4 only)
	11	Reserved	Reserved
	12	DFI_IS_MRW	Asserted when a MRW command is sent by the controller
	13	DFI_IS_MRR	Asserted when a MRR command is sent by the controller
	14	DFI_IS_REFPB	Asserted when a refresh per bank command is sent by the controller
	15	DFI_IS_REFAB	Asserted when a refresh all banks command is sent by the controller
	16	DFI_IS_MPC	Asserted when an MPC command is sent by the controller (LPDDR4 only)

Table 63. Signals monitored at performance logging interface (provided by DDRCTRL)

-	N[0]	Signal name	Description
DDRC recorded DRAM commands	32	PERF_OP_IS_ACT	Asserted for every DRAM activate command logged by the controller
	33	PERF_OP_IS_RD	Asserted for every DRAM read command logged by the controller
	34	PERF_OP_IS_WR	Asserted for every DRAM write command logged by the controller
	35	PERF_OP_IS_MWR	Asserted for every DRAM masked write command logged by the controller
	36	PERF_OP_IS_REF	Asserted for every DRAM refresh command logged by the controller
	37	PERF_OP_IS_CRIT_REF	Asserted for every critical DRAM refresh command logged by the controller
	38	PERF_OP_IS_SPEC_REF	Asserted for every speculative DRAM refresh command logged by the controller
	39	PERF_OP_IS_ZQCAL	Asserted for every ZQCS or ZQCL calibration command logged by the controller
	40	PERF_OP_IS_ENTER_POWDN	Asserted for every entry into DRAM power-down mode
	41	PERF_OP_IS_ENTER_SELFREF	Asserted for every entry into DRAM self-refresh mode
	42	PERF_OP_IS_PRE	Asserted for every precharge command logged by the controller
	43	PERF_OP_IS_PRE_FOR_RDWR	Asserted when the precharge command is associated to a read or write command
	44	PERF_OP_IS_PRE_FOR _OTHERS	Asserted when the precharge command is due to requests other than read or write
	45	PERF_OP_IS_RD_ACTIVATE	Asserts for every read activate that is issued
DDRCTRL p.i. (performance indicators)	48	PERF_HPR_REQ_WITH _NOCREDIT	Asserted when there is a high priority read request (from XPI to port arbiter) which is not served due to no available credit
	49	PERF_LPR_REQ_WITH _NOCREDIT	Asserted when there is a low priority read request (from XPI to port arbiter) which is not served due to no available credit
	50	PERF_HPR_XACT_WHEN _CRITICAL	Asserted for every high priority read transaction that is scheduled when the high priority queue is in critical state
	51	PERF_LPR_XACT_WHEN _CRITICAL	Asserted for every low priority read transaction that is scheduled when the low priority queue is in critical state
	52	PERF_WR_XACT_WHEN _CRITICAL	Asserted for every high priority write transaction that is scheduled when the write queue is in critical state
	53	PERF_RDWR_TRANSITIONS	Asserted for every read to write and write to read bus turnaround
	54	PERF_WAR_HAZARD	Asserted for every write after read collision that happens in the DDRC
	55	RAW_HAZARD	Asserted for every read after write collision that happens in the DDRC
	56	PERF_WAW_HAZARD	Asserted for every write after write collision that happens in the DDRC
	57	Reserved	Reserved
	58	PERF_RANK	Gives the rank number for every scheduled command through the DDRC.

Table 63. Signals monitored at performance logging interface (provided by DDRCTRL) (continued)

-	N[0]	Signal name	Description
DDRCTRL p.i	59	PERF_READ_BYPASS	Asserts for every read command that is sent through the bypass path
	60	PERF_ACT_BYPASS	Asserts for every activate command that is sent through the bypass path
	61	PERF_WINDOW_LIMIT_ REACHED_RD	Indicates that at least one RD CAM entry reaches visible window limit at this cycle. Performance feature to avoid extreme starvation.
	62	PERF_WINDOW_LIMIT_ REACHED_WR	Indicates that at least one WR CAM entry reaches visible window limit at this cycle. Performance feature to avoid extreme starvation.
	63	NO_EVENT	Signal number to be selected for event counters that are unused in the DDRPERFM module

15.4.2 DDRPERFM DFI command decoder

The DFI command decoder is implemented to extract specific events decoding the DFI commands sent by controller to PHY. The inputs are:

- DDR3/4: dfi_address, dfi_cs for all protocols, dfi_ras_n dfi_cas_n, dfi_we_n, dfi_bank
- DDR4 only: dfi_act_n, dfi_bg

The DFI command decoder has a selector (dram_type) which defines the DRAM type attached to the device. It is used to select the DFI decoder depending on the DRAM protocol used.

The following table lists the events generated by DFI command address decoder (concerned filters are highlighted in gray).

Table 64. Events generated by the DFI command address decoder

Event name	DRAM type	Bank filter	BG filter	Comment
DFI_IS_MRW	LPDDR4	No	No	Asserted when a mode register write (MRW) commands is sent by the controller: MRS in DDR3/4
	LPDDR3	No	No	
	DDR4	No	No	
	DDR3	No	No	
DFI_IS_MRR	LPDDR4	No	No	Asserted when a mode register read (MRR) commands is sent by the controller: Does not exist in DDR3/4
	LPDDR3	No	No	
DFI_IS_MPC	LPDDR4	No	No	Asserted when a multi-purpose command (MPC) is sent by the controller. Only for LPDDR4.
DFI_IS_REFPB	LPDDR4	Yes	No	Asserted when a refresh per bank command is sent by the controller. For LPDDR3, bank to refresh is defined in DRAM round robin algorithm.
	LPDDR3	No	No	
DFI_IS_REFAB	LPDDR4	No	No	Asserted when a refresh all banks command is sent by the controller. For DDR3/4, the refresh addressing is generated by the internal refresh controller.
	LPDDR3	No	No	
	DDR4	No	No	
	DDR3	No	No	

Table 64. Events generated by the DFI command address decoder (continued)

Event name	DRAM type	Bank filter	BG filter	Comment
DFI_IS_PREPB	LPDDR4	Yes	No	Asserted when an PRE per bank command is sent by the controller.
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	
DFI_IS_PREAB	LPDDR4	No	No	Asserted when an PRE all banks command is sent by the controller.
	LPDDR3	No	No	
	DDR4	No	No	
	DDR3	No	No	
DFI_IS_ACT	LPDDR4	Yes	No	Asserted when an ACTIVATE command is sent by the controller.
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	
DFI_IS_WR	LPDDR4	Yes	No	Asserted when a WR command is sent by the controller (AP bit ignored).
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	
DFI_IS_WRA	LPDDR4	Yes	No	Asserted when a WR with AP command is sent by the controller.
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	
DFI_IS_MWR	LPDDR4	Yes	No	Asserted when a masked write command is sent by the controller (AP bit ignored, LPDDR4 only)
DFI_IS_MWRA	LPDDR4	Yes	No	Asserted when a masked write command with AP is sent by the controller (LPDDR4 only)
DFI_IS_RD	LPDDR4	Yes	No	Asserted when a RD command is sent by the controller (AP bit is ignored)
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	
DFI_IS_RDA	LPDDR4	Yes	No	Asserted when a RD command with AP is sent by the controller
	LPDDR3	Yes	No	
	DDR4	Yes	Yes	
	DDR3	Yes	No	

15.5 DDRPERFM registers

This section presents the registers for controlling the performance monitor.

15.5.1 DDRPERFM control register (DDRPERFM_CTRL)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP	START
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STOP**:

Writing 1 stops all counters. Writing 0 has no effect. Writing 1 when counters are stopped has not effect.

Bit 0 **START**:

Writing 1 starts all counters. All internal counters and status bits are cleared before events are monitored. Writing 0 has no effect. Writing 1 when counters are running has not effect.

15.5.2 DDRPERFM interrupt mask register (DDRPERFM_IMSK)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTRMSK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **INTRMSK**:

0: No effect

1: DDRPERFM interruption signal masked.

15.5.3 DDRPERFM interrupt clear register (DDRPERFM_ICLR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTRSET	INTRCLR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **INTRSET**:

0: No effect

1: Set the interruption signal when asserted.

Bit 0 **INTRCLR**:

0: No effect

1: Clear the interruption signal when asserted.

Note: The counters must be cleared first (to remove the interruption condition) before the interruption is cleared.

15.5.4 DDRPERFM counter clear register (DDRPERFM_CLR)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TNTCLR	CNTCLR[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TNTCLR**:

0: No effect

1: Clear free running counter and associated status bits (timeout and overflow indicators).

Note: Setting this bit is ignored if BUSY = 1 in DDRPERFM_STATUS.

Bits 7:0 **CNTCLR[7:0]**:

0: No effect

1: Clear corresponding counters and associated status bits (timeout and overflow indicators).

Note: Setting this bit is ignored if BUSY = 1 in DDRPERFM_STATUS.

15.5.5 DDRPERFM configuration register 0 (DDRPERFM_CFG0)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SEL_EVENT3[5:0]						Res.	Res.	SEL_EVENT2[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SEL_EVENT1[5:0]						Res.	Res.	SEL_EVENT0[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **SEL_EVENT3[5:0]**: Selection of the performance logging signal to be monitored by the event counter 3

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **SEL_EVENT2[5:0]**: Selection of the performance logging signal to be monitored by the event counter 2

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **SEL_EVENT1[5:0]**: Selection of the performance logging signal to be monitored by the event counter 1

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **SEL_EVENT0[5:0]**: Selection of the performance logging signal to be monitored by the event counter 0

15.5.6 DDRPERFM configuration register 1 (DDRPERFM_CFG1)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SEL_EVENT7[5:0]						Res.	Res.	SEL_EVENT6[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SEL_EVENT5[5:0]						Res.	Res.	SEL_EVENT4[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **SEL_EVENT7[5:0]**: Selection of the performance logging signal to be monitored by the event counter 7

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **SEL_EVENT6[5:0]**: Selection of the performance logging signal to be monitored by the event counter 6

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **SEL_EVENT5[5:0]**: Selection of the performance logging signal to be monitored by the event counter 5

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **SEL_EVENT4[5:0]**: Selection of the performance logging signal to be monitored by the event counter 4

15.5.7 DDRPERFM configuration register 2 (DDRPERFM_CFG2)

Address offset: 0x0018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FILT_POL3[2:0]			FILT_RBG3[4:0]					FILT_POL2[2:0]			FILT_RBG2[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FILT_POL1[2:0]			FILT_RBG1[4:0]					FILT_POL0[2:0]			FILT_RBG0[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FILT_POL3[2:0]**: Selection of the performance logging signal to be monitored by the event counter 3

Filtering policy for the event counter 3 (refer to *Table 65*).

Bits 28:24 **FILT_RBG3[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 3 based on FILT_POL3 value

Bit 4: Define the rank value when rank filtering is activated. Value 0 means that only commands applied on Rank0 are counted. Value 1 means that only commands applied on rank1 are counted.

Bits 3:0: Define drank group/bank address/ used for filtering

When filtering is activated:

- For DDR4, the FILT_RBGx[3:2] bits correspond to bank group, and FILT_RBGx[1:0] correspond to bank address.
- For LPDDR3/4, the DDR3 bits FILT_RBGx[2:0] bits correspond to bank address, and FILT_RBG3 is unused.

Bits 23:21 **FILT_POL2[2:0]**: Filtering policy for the event counter 2. (refer to *Table 65*)

Bits 20:16 **FILT_RBG2[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 2 based on FILT_POL2 value

Bits 15:13 **FILT_POL1[2:0]**: Filtering policy for the event counter 1 (refer to *Table 65*)

Bits 12:8 **FILT_RBG1[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 1 based on FILT_POL1 value

Bits 7:5 **FILT_POL0[2:0]**: Filtering policy for the event counter 0 (refer to *Table 65*)

Bits 4:0 **FILT_RBG0[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 0 based on FILT_POL0 value

Table 65. Filtering policy

FILT_POLx	Rank filtering	Bank group filtering[1]	Bank filtering	comment
000	No	No	No	No filtering is applied. The counter counts all events whatever rank, bank group and bank address.
001	No	No	Yes	Filtering based on FILT_RBGx value
010	No	Yes	No	Filtering based on FILT_RBGx value
011	No	Yes	Yes	Filtering based on FILT_RBGx value
100	Yes	No	No	Filtering based on FILT_RBGx value
101	Yes	No	Yes	Filtering based on FILT_RBGx value
110	Yes	Yes	No	Filtering based on FILT_RBGx value
111	Yes	Yes	Yes	Filtering based on FILT_RBGx value

1. Only relevant for DDR4.

15.5.8 DDRPERFM configuration register 3 (DDRPERFM_CFG3)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FILT_POL7[2:0]			FILT_RBG7[4:0]					FILT_POL6[2:0]			FILT_RBG6[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FILT_POL5[2:0]			FILT_RBG5[4:0]					FILT_POL4[2:0]			FILT_RBG4[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FILT_POL7[2:0]**: Filtering policy for the event counter 7 (refer to *Table 65*)

Bits 28:24 **FILT_RBG7[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 7 based on FILT_POL7 value

Bits 23:21 **FILT_POL6[2:0]**: Filtering policy for the event counter 6 (refer to *Table 65*)

Bits 20:16 **FILT_RBG6[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 6 based on FILT_POL6 value

Bits 15:13 **FILT_POL5[2:0]**: Filtering policy for the event counter 5 (refer to *Table 65*)

Bits 12:8 **FILT_RBG5[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 5 based on FILT_POL5 value

Bits 7:5 **FILT_POL4[2:0]**: Filtering policy for the event counter 4 (refer to *Table 65*)

Bits 4:0 **FILT_RBG4[4:0]**: Configuration of the rank/bank/bank group filtering for the event counter 4 based on FILT_POL4 value

15.5.9 DDRPERFM configuration register 4 (DDRPERFM_CFG4)

Address offset: 0x0020

Reset value: 0xFFFF FFFF

Timeout value for the time counter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME_OUT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME_OUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIME_OUT[31:0]**: Timeout value for the time counter
When this value is reached, the DDRPERFM automatically stops: All counters are frozen. The status bit TCNT_TIMOUT is asserted. The (maskable) interrupt signal is asserted.

15.5.10 DDRPERFM event counter enable register (DDRPERFM_CFG5)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVCNT_EN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EVCNT_EN[7:0]**: Event counter enable
When bit <i> is set, the corresponding counter is enabled.
Note: Enable must be written when counters are not running.

15.5.11 DDRPERFM DRAM information register (DDRPERFM_DRAMINF)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_TYPE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **DRAM_TYPE[1:0]:** DRAM information used for DFI decoding
00: LPDDR4
01: LPDDR3
10: DDR4
11: DDR3

15.5.12 DDRPERFM event counter 0 register (DDRPERFM_EVCNT0)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT0[31:0]**:Counter value for event 0

15.5.13 DDRPERFM event counter 1 register (DDRPERFM_EVCNT1)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT1[31:0]**:Counter value for event 1

15.5.14 DDRPERFM event counter 2 register (DDRPERFM_EVCNT2)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT2[31:0]**: Counter value for event 2

15.5.15 DDRPERFM event counter 3 register (DDRPERFM_EVCNT3)

Address offset: 0x004C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT3[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT3[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT3[31:0]**: Counter value for event 3

15.5.16 DDRPERFM event counter 4 register (DDRPERFM_EVCNT4)

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT4[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT4[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT4[31:0]**: Counter value for event 4

15.5.17 DDRPERFM event counter 5 register (DDRPERFM_EVCNT5)

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT5[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT5[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT5[31:0]**: Counter value for event 5

15.5.18 DDRPERFM event counter 6 register (DDRPERFM_EVCNT6)

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT6[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT6[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT6[31:0]**: Counter value for event 6

15.5.19 DDRPERFM event counter 7 register (DDRPERFM_EVCNT7)

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVCNT7[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVCNT7[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVCNT7[31:0]**: Counter value for event 7

15.5.20 DDRPERFM time counter register (DDRPERFM_TCNT)

Address offset: 0x0060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TCNT[31:0]**: Time counter value

Must only be read when the PERFM is stopped.

15.5.21 DDRPERFM status register (DDRPERFM_STATUS)

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCNT_OVFL
r															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVCNT7_OVFL	EVCNT6_OVFL	EVCNT5_OVFL	EVCNT4_OVFL	EVCNT3_OVFL	EVCNT2_OVFL	EVCNT1_OVFL	EVCNT0_OVFL
								r	r	r	r	r	r	r	r

Bit 31 **BUSY**: Real-time busy indicator

This bit is asserted as long as counters are running, and deasserted when a stop condition occurs (software controlled stop command or counter overflow).

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **TCNT_OVFL**:

When read to 1, the DDRPERFM is stopped due to time counter reaching the timeout value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **EVCNT7_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 7 overflow.

Bit 6 **EVCNT6_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 6 overflow.

Bit 5 **EVCNT5_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 5 overflow.

Bit 4 **EVCNT4_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 4 overflow.

Bit 3 **EVCNT3_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 3 overflow.

Bit 2 **EVCNT2_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 2 overflow.

Bit 1 **EVCNT1_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 1 overflow.

Bit 0 **EVCNT0_OVFL**:

When read to 1, the DDRPERFM is stopped due to event counter 0 overflow.

15.5.22 DDRPERFM version register (DDRPERFM_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** DDRPERFM major revision

Bits 3:0 **MINREV[3:0]:** DDRPERFM minor revision

15.5.23 DDRPERFM identification register (DDRPERFM_ID)

Address offset: 0x3F8

Reset value: 0x0014 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** DDRPERFM unique identification

15.5.24 DDRPERFM magic ID register (DDRPERFM_SID)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Magic ID for automatic DDRPERFM discovery

15.5.25 DDRPERFM register map

Table 66. DDRPERFM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DDRPERFM_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP	START
	Reset value																															0	0
0x004	DDRPERFM_IMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTRMSK
	Reset value																																0
0x008	DDRPERFM_ICLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTRSET	INTRCLR
	Reset value																															0	0
0x00C	DDRPERFM_CLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCNTCLR	CNTCLR[7:0]							
	Reset value																								0	0	0	0	0	0	0	0	0
0x010	DDRPERFM_CFG0	Res.	Res.	SEL_EVENT3[5:0]						Res.	Res.	SEL_EVENT2[5:0]						Res.	Res.	SEL_EVENT1[5:0]						Res.	Res.	SEL_EVENT0[5:0]			
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x014	DDRPERFM_CFG1	Res.	Res.	SEL_EVENT7[5:0]						Res.	Res.	SEL_EVENT6[5:0]						Res.	Res.	SEL_EVENT5[5:0]						Res.	Res.	SEL_EVENT4[5:0]			
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x018	DDRPERFM_CFG2	FILT_POL3 [2:0]			FILT_RBG3[4:0]					FILT_POL2 [2:0]			FILT_RBG2[4:0]					FILT_POL1 [2:0]			FILT_RBG1[4:0]				FILT_POL0 [2:0]			FILT_RBG0[4:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DDRPERFM_CFG3	FILT_POL7 [2:0]			FILT_RBG7[4:0]					FILT_POL6 [2:0]			FILT_RBG6[4:0]					FILT_POL5 [2:0]			FILT_RBG5[4:0]				FILT_POL4 [2:0]			FILT_RBG4[4:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DDRPERFM_CFG4	TIME_OUT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x024	DDRPERFM_CFG5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVCNT_EN[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	DDRPERFM_DRAMINF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_TYPE[1:0]
	Reset value																																0
0x02C-0x03C	Reserved	Reserved																															
0x040 + 0x4 * x (x=0 to 7)	DDRPERFM_EVCNTx	EVCNTx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DDRPERFM_TCNT	TCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064-0x07C	Reserved	Reserved																															
0x080	DDRPERFM_STATUS	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCNT_OVFL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVCNT7_OVFL	EVCNT6_OVFL	EVCNT5_OVFL	EVCNT4_OVFL	EVCNT3_OVFL	EVCNT2_OVFL	EVCNT1_OVFL	EVCNT0_OVFL
	Reset value	0															0								0	0	0	0	0	0	0	0

Table 66. DDRPERFM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084-0x3F0	Reserved	Reserved																															
0x3F4	DDRPERFM_VERR	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	RES	MAJREV[3:0]					MINREV[3:0]				
	Reset value																							0	0	0	0	1	0	0	0	0	0
0x3F8	DDRPERFM_ID	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
0x3FC	DDRPERFM_SID	SID[31:0]																															
	Reset value	1	0	0	1	0	0	1	1	1	0	1	1	0	1	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

16 SRAM configuration controller (RAMCFG)

16.1 RAMCFG introduction

The RAMCFG configures the features of the internal SRAMs (SYSRAM, RETRAM, SRAM1 and BKPSRAM).

16.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt generation
  - Status with failing address
- SRAM hardware erase
- SRAM software erase
- Cyclic redundancy check (CRC-32)

16.3 RAMCFG functional description

16.3.1 Internal SRAM features

Four SRAMs are embedded in the STM32MP21xx devices, each with specific features:

- SYSRAM and SRAM1 are the main SRAMs. BKPSRAM and RETRAM have retention feature for low-power modes even when V_{DD} is off in V_{BAT} mode. See *Section 19: Power control (PWR)* for more details.
- The SYSRAM is partially or fully erased (upper 128 Kbytes erased when allowed by SYSRAM_ERASE option byte) when a system reset occurs. The RETRAM is erased when a system reset occurs if SRAMHWERDIS in RAMCFG_RETRAMCR is not set.
- SRAM1 and BKPSRAM are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a backup domain supply power-on (see *Section 64: Tamper and backup registers (TAMP)*).
- The RAMCFG embeds the registers related to the internal SRAMs ECC, CRC and software erase.

The tables below summarize the features supported by each internal SRAM.

Table 67. Internal SRAM features

SRAM feature	SYSRAM	RETRAM	SRAM1	BKPSRAM
	256 Kbytes	128 Kbytes	64 Kbytes	8 Kbytes
LPBAM Stop/LP-Stop	-	X	X	-
Optional retention in V_{BAT} mode	-	X	-	X
Erase with tamper detection	-	-	X	X
Hardware erase	X	X	-	-
Software erase	X	X	X	X
ECC	-	X	-	X
CRC	-	X	-	-

Table 68. Internal SRAM hardware erase feature

SRAM (size in Kbytes)	Conditional hardware erase	Condition
SYSRAM low bank (128)	No	Erase done on each system reset
SYSRAM high bank (128)	Yes	Erase on system reset only once SYSRAM_ERASE in OTP HCONF1 is fused to 1
RETRAM (128)	Yes	Erase on system reset. Can be disabled by SRAMHWERDIS in RAMCFG_RETRAMCR.

16.3.2 Error code correction (RETRAM, BKPSRAM)

The ECC is supported by RETRAM and BKPSRAM.

Seven ECC bits are added per 32 bits of SRAM, allowing a 2-bit error detection and 1-bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then writing the word again with the new byte/half-word value. The ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is two AHB clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM, two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

Single- and double-ECC errors

When a single error is detected, it is automatically corrected and the SEC bit is set in RAMCFG_xISR. An interrupt is generated if enabled by SEIE in RAMCFG_xIER. The failing address is stored in RAMCFG_xSEAR if ALE is set in RAMCFG_xCR.

Caution: Single errors are not reported if the SEC flag is not cleared in time when a new error happens.

When a double error is detected, the DED bit is set in RAMCFG_xISR. An interrupt is generated if enabled by DEIE in RAMCFG_xIER. The failing address is stored in RAMCFG_xDEAR if ALE is set in RAMCFG_xCR.

Caution: Double errors are not reported if the DED flag is not cleared. Double errors are not corrected, and must be managed by exception.

The ECC can be activated / deactivated by executing the following software sequence:

1. Write 0xAE in RAMCFG_xECCKEYR.
2. Write 0x75 in RAMCFG_xECCKEYR.
3. Write 1 in the ECCE bit of RAMCFG_xCR to activate the ECC (write 0 in this bit to deactivate the ECC).

RETRAM ECC specific management

The ECC must be enabled on RETRAM to allow the CPU2 boot when exiting Standby modes in wake-up event mode (ignored in legacy boot mode). If a double error is detected during the wake-up phase, an application reset may be generated (reset always generated if the CPU2 is the trusted domain). Refer to *Section 20: Reset and clock control (RCC)* for more details.

This signal (combined with CRC check error) is also routed to the EXTI2 in order to wake up the CPU1.

16.3.3 Cyclic redundancy check (RETRAM)

The hardware CRC (32-bit CRC P(x) = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$) is supported by RETRAM.

The CRC module may be used to calculate a signature of part of the memory content (instruction code in RETRAM for the CPU2) before jumping in low-power mode. When exiting low-power mode, the CRC can be used to check, without any CPU intervention, that the value of the memory is not modified/altered.

The CRC check can be launched by software or by hardware when exiting low-power modes as follows:

- For the RETRAM, the CRC engine (if check is enabled before entering low-power mode) starts when the system wakes up from Standby mode.

Refer to *Section 19: Power control (PWR)* for more details.

The following software sequence must be executed to calculate the signature of the memory content:

1. Write 0x0 in CRCC[1:0] of RAMCFG_RETRAMCCR1 to deactivate the CRC.
2. Write 1 in SRAMHWERDIS of RAMCFG_RETRAMCR to disable the hardware erase.
3. Select the buffer size to take into account during the CRC computation by setting CRCBS[2:0] in RAMCFG_RETRAMCCR1.
4. Write 0x1 in CRCC[1:0] of RAMCFG_RETRAMCCR1 to activate the CRC.
5. Write 1 in CRCCS of RAMCFG_RETRAMCCR2 to start the CRC computation.

The calculated signature is stored in RAMCFG_RETRAMCCSR when CRCEOC is set in RAMCFG_RETRAMCSR.

The following software sequence must be executed to check the memory content to prepare

the CRC signature used while exiting a low-power mode:

1. Write 0x0 in CRCC[1:0] of RAMCFG_RETRAMCCR1 to deactivate the CRC.
2. Write 1 in SRAMHWERDIS of RAMCFG_RETRAMCR to disable the hardware erase.
3. Select the buffer size to take into account during CRC computation by setting CRCBS[2:0] in RAMCFG_RETRAMCCR1.
4. Write 0x1 in CRCC[1:0] of RAMCFG_RETRAMCCR1 to activate the CRC.
5. Write 1 in CRCCS of RAMCFG_RETRAMCCR2 to start the CRC computation.
6. Wait until CRCEOC is set in RAMCFG_RETRAMCSR.
7. Read calculated signature in RAMCFG_RETRAMCCSR.
8. Write calculated signature in RAMCFG_RETRAMCRSR.
9. Write 1 in CRCFC of RAMCFG_RETRAMCCR2 to clear the CRCEOC flag.
10. Write 0x2 in CRCC[1:0] of RAMCFG_RETRAMCCR1 to enable the CRC computation and signature check.

When exiting a low-power mode, the CRC calculates the signature of the memory content and compares the signature against the precalculated signature.

If no error is detected during signature check, CRCSCS and CRCEOC are set in RAMCFG_RETRAMCSR.

RETRAM CRC specific management

The CRC computation and signature check must be enabled on RETRAM to allow the CPU2 boot when exiting Standby mode in wake-up event mode (ignored in the legacy boot mode). If the CRC check fails, the CPU2 is not allowed to boot and an application reset or a notification can be generated (reset always generated if the CPU2 is the trusted CPU). The CRC check is ignored in the legacy boot mode. See *Section 20: Reset and clock control (RCC)* for more details on the RETRAM CRC error system reset enable.

This signal (combined with the ECC DED) is also routed to the EXTI2 in order to wake up the CPU1.

16.4 RAMCFG low-power modes

Table 69. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit Sleep mode.
Stop	The content of RAMCFG registers is kept.
Standby	The RAMCFG is powered down. Only few register bits are kept in Standby (see bit description for more details).

16.5 RAMCFG interrupts

Four interrupts are generated internally (all maskable by software) and a global interrupt request is exported to the interrupt controller. The table below gives the list of RAMCFG interrupt requests.

Table 70. RAMCFG interrupt requests

Interrupt acronym	Memory	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
RAMCFG	RETRAM	ECC single error detection and correction	SEC	SEIE	Write 1 in CSEC
		ECC double error detection	DED	DEIE	Write 1 in CDED
	BKPSRAM	ECC single error detection and correction	SEC	SEIE	Write 1 in CSEC
		ECC double error detection	DED	DEIE	Write 1 in CDED

1. All these bits are in RAMCFG_RETRAMISR.

16.6 RAMCFG registers

The RAMCFG registers can be accessed in word, half-word, and byte format, unless explicitly specified differently.

16.6.1 RAMCFG SYSRAM control register (RAMCFG_SYSRAMCR)

Address offset: 0x000

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_SYSRAMERASEKEY. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

16.6.2 RAMCFG SYSRAM interrupt status register (RAMCFG_SYSRAMISR)

Address offset: 0x008

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation ongoing
1: Erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection. Refer to Table 67: Internal SRAM features.

Bits 7:0 Reserved, must be kept at reset value.

16.6.3 RAMCFG SYSRAM erase key register (RAMCFG_SYSRAMERKEYR)

Address offset: 0x028

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_SYSRAMCR.

a) Write 0xCA into ERASEKEY[7:0].
b) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.4 RAMCFG SRAM1 control register (RAMCFG_SRAM1CR)

Address offset: 0x080

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase.
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_MCURAM1ERASEKEY. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

16.6.5 RAMCFG SRAM1 interrupt status register (RAMCFG_SRAM1ISR)

Address offset: 0x088

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation
0: No erase operation ongoing
1: Erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection. Refer to Table 67: Internal SRAM features.

Bits 7:0 Reserved, must be kept at reset value.

16.6.6 RAMCFG SRAM1 erase key register (RAMCFG_SRAM1ERKEYR)

Address offset: 0x0A8

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_MCURAM1CR.

a) Write 0xCA into ERASEKEY[7:0].

b) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.7 RAMCFG RETRAM control register (RAMCFG_RETRAMCR)

Address offset: 0x180

Reset value: 0x0000 0000

See individual bits for reset condition.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SRAMHWERDIS	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
			rw				rs				rw				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SRAMHWERDIS:** SRAM hardware erase disable

0: Erase operation done after a system reset

1: No erase operation done after a system reset

Note: This bit is reset on a reset only by VSW POR and VSWRST.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase.
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_RETRAMERASEKEY. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing
Note: This bit is reset on any system reset.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE:** Address latch enable
0: Failing address not stored in the RETRAMECC single/double error address registers
1: Failing address stored in the RETRAMECC single/double error address registers
Note: This bit is reset on any system reset.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable
When set, this bit can be cleared by software only after writing the unlock sequence in RAMCFG_RETRAMECCKEYR.
0: ECC disabled
1: ECC enabled
Note: This bit is reset on an application reset.

16.6.8 RAMCFG RETRAM interrupt enable register (RAMCFG_RETRAMIER)

Address offset: 0x184

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEIE	SEIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEIE:** ECC double error interrupt enable
0: Double error interrupt disabled
1: Double error interrupt enabled

Bit 0 **SEIE:** ECC single error interrupt enable
0: Single error interrupt disabled
1: Single error interrupt enabled

16.6.9 RAMCFG RETRAM interrupt status register (RAMCFG_RETRAMISR)

Address offset: 0x188

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEC
							r							r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation ongoing
1: Erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection. Refer to Table 67: Internal SRAM features.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DED:** ECC double error detected

0: No double error
1: Double error detected

Bit 0 **SEC:** ECC single error corrected

0: No single error
1: Single error corrected

16.6.10 RAMCFG RETRAM ECC single error address register (RAMCFG_RETRAMSEAR)

Address offset: 0x18C

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ESEA[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **ESEA[14:0]:** ECC single error address

When ALE is set in RAMCFG_RETRAMCR, this bitfield is updated with the word-address corresponding to the ECC single error.

16.6.11 RAMCFG RETRAM ECC double error address register (RAMCFG_RETRAMDEAR)

Address offset: 0x190

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EDEA[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **EDEA[14:0]:** ECC double error address

When ALE is set in RAMCFG_RETRAMCR, this bitfield is updated with the word-address corresponding to the ECC double error.

16.6.12 RAMCFG RETRAM interrupt clear flag register (RAMCFG_RETRAMICR)

Address offset: 0x194

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEC
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED:** Clear of ECC double error detected

Writing 1 to this flag clears DED in RAMCFG_RETRAMISR.

Bit 0 **CSEC:** Clear of ECC single corrected

Writing 1 to this flag clears SEC in RAMCFG_RETRAMISR.

16.6.13 RAMCFG RETRAM ECC key register (RAMCFG_RETRAMECCKEYR)

Address offset: 0x1A4

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]:** ECC write protection key

The following steps are required to unlock the write protection of ECCE in RAMCFG_RETRAMCR.
a) Write 0xAE into ECCKEY[7:0].
b) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.14 RAMCFG RETRAM erase key register (RAMCFG_RETRAMERKEYR)

Address offset: 0x1A8

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_RETRAMCR.
a) Write 0xCA into ERASEKEY[7:0].
b) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.15 RAMCFG RETRAM CRC control register 1 (RAMCFG_RETRAMCCR1)

Address offset: 0x1B0

Reset value: 0x0000 0070

Not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCBS[2:0]			Res.	Res.	CRCC[1:0]	
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CRCBS[2:0]**: CRC buffer size
This bitfield is used to program the buffer size to take into account during CRC computation. Starting from RETRAM start address.
0x0: 16 Kbytes
0x1: 32 Kbytes
0x2: 48 Kbytes
.....
0x7: 128 Kbytes (default)

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **CRCC[1:0]:** CRC configuration.
00: CRC disabled
01: CRC computation enabled. The calculated signature is stored in RAMCFG_RETRAMCCSR when the CRCEOC flag is set.
10: CRC computation and signature check enabled. The status of the signature check is available in RAMCFG_RETRAMCSR when the CRCEOC flag is set.
11: Reserved

16.6.16 RAMCFG RETRAM CRC control register 2 (RAMCFG_RETRAMCCR2)

Address offset: 0x1B4

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCCS
															rw

Bit 31 **CRCFC:** CRC flags clear.
Setting this bit clears the CRC status flags in RAMCFG_RETRAMCSR.
0: CRC flags are not affected.
1: CRC flags are cleared in RAMCFG_RETRAMCSR.
Note: This bit is automatically cleared by hardware.

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **CRCCS**: CRC computation start
Setting this bit starts the CRC computation. This bit is automatically cleared by hardware at the end of the CRC computation phase.
0: CRC computation not launched
1: CRC computation launched

16.6.17 RAMCFG RETRAM CRC reference signature register (RAMCFG_RETRAMCRSR)

Address offset: 0x1B8

Reset value: 0x0000 0000

Not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCRS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCRS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCRS[31:0]:** CRC reference signature.
This bitfield is used to write the reference signature (precalculated signature) used for CRC signature comparison check.

16.6.18 RAMCFG RETRAM CRC status register (RAMCFG_RETRAMCSR)

Address offset: 0x1BC

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCSCS	CRCEOC
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CRCSCS:** CRC signature check status flag

This bit is set when the computed signature matches the reference signature stored in RAMCFG_RETRAMCRSR. This bit is valid only when the CRCEOC flag is set.

0: Fault detected is SRAM data. Calculated signature does not match the reference signature.

1: SRAM data is correct. Calculated signature matches the reference signature.

Bit 0 **CRCEOC:** CRC end-of-computation flag

0: Computation ongoing

1: Computation done

16.6.19 RAMCFG RETRAM CRC calculated signature register (RAMCFG_RETRAMCCSR)

Address offset: 0x1C0

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCCS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCCS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CRCCS[31:0]:** CRC calculated signature

When CRCEOC is set in RAMCFG_RETRAMCSR, this bitfield is updated with the calculate signature representing the content of the SRAM buffer to compute.

16.6.20 RAMCFG BKPSRAM control register (RAMCFG_BKPSRAMCR)

Address offset: 0x380

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
							rs				rw				rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase.
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_BKPSRAMERASEKEY. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE:** Address latch enable
0: Failing address not stored in the BKPSRAMECC single/double error address registers
1: Failing address stored in the BKPSRAMECC single/double error address registers

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable
When set, this bit can be cleared by software only after writing the unlock sequence in RAMCFG_BKPSRAMECCKEYR.
0: ECC disabled
1: ECC enabled

16.6.21 RAMCFG BKPSRAM interrupt enable register (RAMCFG_BKPSRAMIER)

Address offset: 0x384

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEIE	SEIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEIE:** ECC double-error interrupt enable
0: Double-error interrupt disabled
1: Double-error interrupt enabled

Bit 0 **SEIE:** ECC single-error interrupt enable
0: Single-error interrupt disabled
1: Single-error interrupt enabled

16.6.22 RAMCFG BKPSRAM interrupt status register (RAMCFG_BKPSRAMISR)

Address offset: 0x388

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEC
							r							r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation ongoing
1: Erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection. Refer to Table 67: Internal SRAM features.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DED:** ECC double error detected

0: No double error
1: Double error detected

Bit 0 **SEC:** ECC single error corrected

0: No single error
1: Single error detected and corrected

16.6.23 RAMCFG BKPSRAM ECC single error address register (RAMCFG_BKPSRAMEEAR)

Address offset: 0x38C

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ESEA[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **ESEA[10:0]:** ECC single error address

When ALE is set in RAMCFG_BKPSRAMCR, this field is updated with the word-address corresponding to the ECC single error.

16.6.24 RAMCFG BKPSRAM ECC double error address register (RAMCFG_BKPSRAMDEAR)

Address offset: 0x390

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EDEA[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **EDEA[10:0]:** ECC double error address

When ALE is set in RAMCFG_BKPSRAMCR, this bitfield is updated with the word-address corresponding to the ECC double error.

16.6.25 RAMCFG BKPSRAM interrupt clear flag register (RAMCFG_BKPSRAMICR)

Address offset: 0x394

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEC
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED:** Clear of ECC double error detected

Writing 1 to this flag clears DED in RAMCFG_BKPSRAMISR.

Bit 0 **CSEC:** Clear of ECC single corrected

Writing 1 to this flag clears SEC in RAMCFG_BKPSRAMISR.

16.6.26 RAMCFG BKPSRAM ECC key register (RAMCFG_BKPSRAMECCKEYR)

Address offset: 0x3A4

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]:** ECC write protection key

The following steps are required to unlock the write protection of ECCE in RAMCFG_BKPSRAMCR.
a) Write 0xAE into ECCKEY[7:0].
b) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.27 RAMCFG BKPSRAM erase key register (RAMCFG_BKPSRAMERKEYR)

Address offset: 0x3A8

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_BKPSRAMCR.
a) Write 0xCA into ERASEKEY[7:0].
b) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

16.6.28 RAMCFG register map

Table 71. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**RAMCFG_SYSRAMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x004	Reserved	Reserved																															
0x008	**RAMCFG_SYSRAMISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x00C-0x024	Reserved	Reserved																															
0x028	**RAMCFG_ SYSRAMERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C-0x07C	Reserved	Reserved																															
0x080	**RAMCFG_SRAM1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x084	Reserved	Reserved																															
0x088	**RAMCFG_SRAM1ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x08C-0x0A4	Reserved	Reserved																															
0x0A8	**RAMCFG_ SRAM1ERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0AC-0x17C	Reserved	Reserved																															
0x180	**RAMCFG_RETRAMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMHWERDIS	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																				0				0				0				0
0x184	**RAMCFG_RETRAMIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEIE	SEIE
	Reset value																															0	0
0x188	**RAMCFG_RETRAMISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEC
	Reset value																								0							0	0

Table 71. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18C	**RAMCFG_ RETRAMSEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESEA[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	**RAMCFG_ RETRAMDEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDEA[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	**RAMCFG_RETRAMICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEC
	Reset value																															0	0
0x198-0x1A0	Reserved	Reserved																															
0x1A4	**RAMCFG_ RETRAMECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1A8	**RAMCFG_ RETRAMERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1AC	Reserved	Reserved																															
0x1B0	**RAMCFG_ RETRAMCCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCBS [2:0]			Res.	Res.	CRCC [1:0]	
	Reset value																										1	1	1			0	0
0x1B4	**RAMCFG_ RETRAMCCR2**	CRCFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCCS
	Reset value	0																															0
0x1B8	**RAMCFG_ RETRAMCRSR**	CRCRS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1BC	**RAMCFG_ RETRAMCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCSCS	CRCEOC
	Reset value																															0	0
0x1C0	**RAMCFG_ RETRAMCCSR**	CRCCS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C4-0x37C	Reserved	Reserved																															
0x380	**RAMCFG_BKPSRAMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				0
0x384	**RAMCFG_BKPSRAMIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEIE	SEIE
	Reset value																															0	0

Table 71. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x388	**RAMCFG_BKPSRAMISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEC
	Reset value																								0							0	0
0x38C	**RAMCFG_ BKPSRAMEEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESEA[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x390	**RAMCFG_ BKPSRAMDEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDEA[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x394	**RAMCFG_BKPSRAMICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEC
	Reset value																															0	0
0x398-0x3A0	Reserved	Reserved																															
0x3A4	**RAMCFG_ BKPSRAMECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3A8	**RAMCFG_ BKPSRAMERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3AC-0x4FC	Reserved	Reserved																															

Refer to *Section 2.4* for the register boundary addresses.

17 Instruction cache (ICACHE)

17.1 ICACHE introduction

The instruction cache (ICACHE) is implemented on the C-AHB code bus of the Cortex®-M33 processor to improve performance when fetching instructions and data from internal and external memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

17.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

17.3 ICACHE implementation

Table 72. ICACHE features

Feature	ICACHE
Number of ways	2
Cache size	16 Kbytes
Cache line width	32 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	32 bits
Data size of AHB slow master2 interface	32 bits

17.4 ICACHE functional description

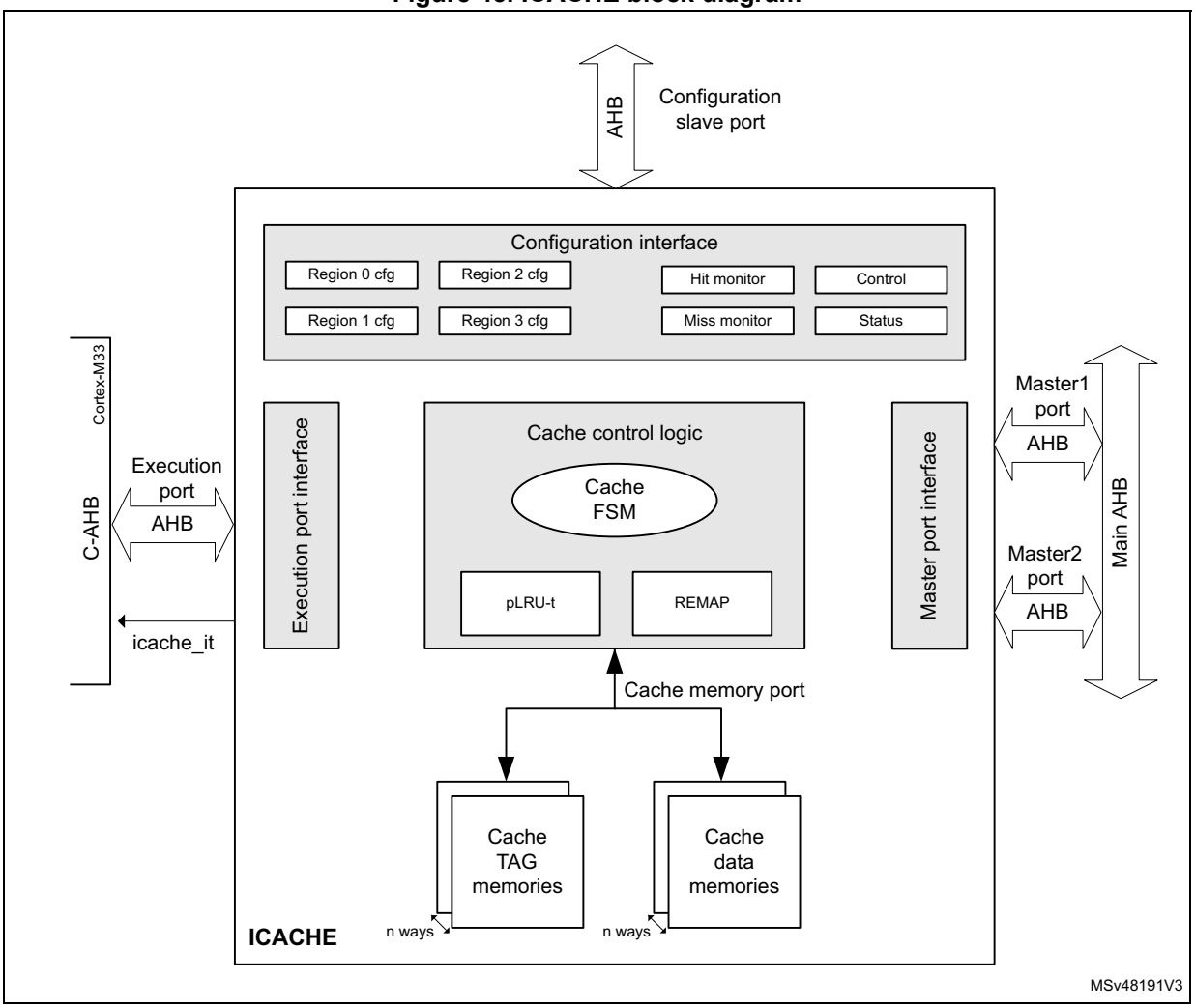
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

17.4.1 ICACHE block diagram

Figure 45. ICACHE block diagram



17.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

17.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 73* for B and W definitions):

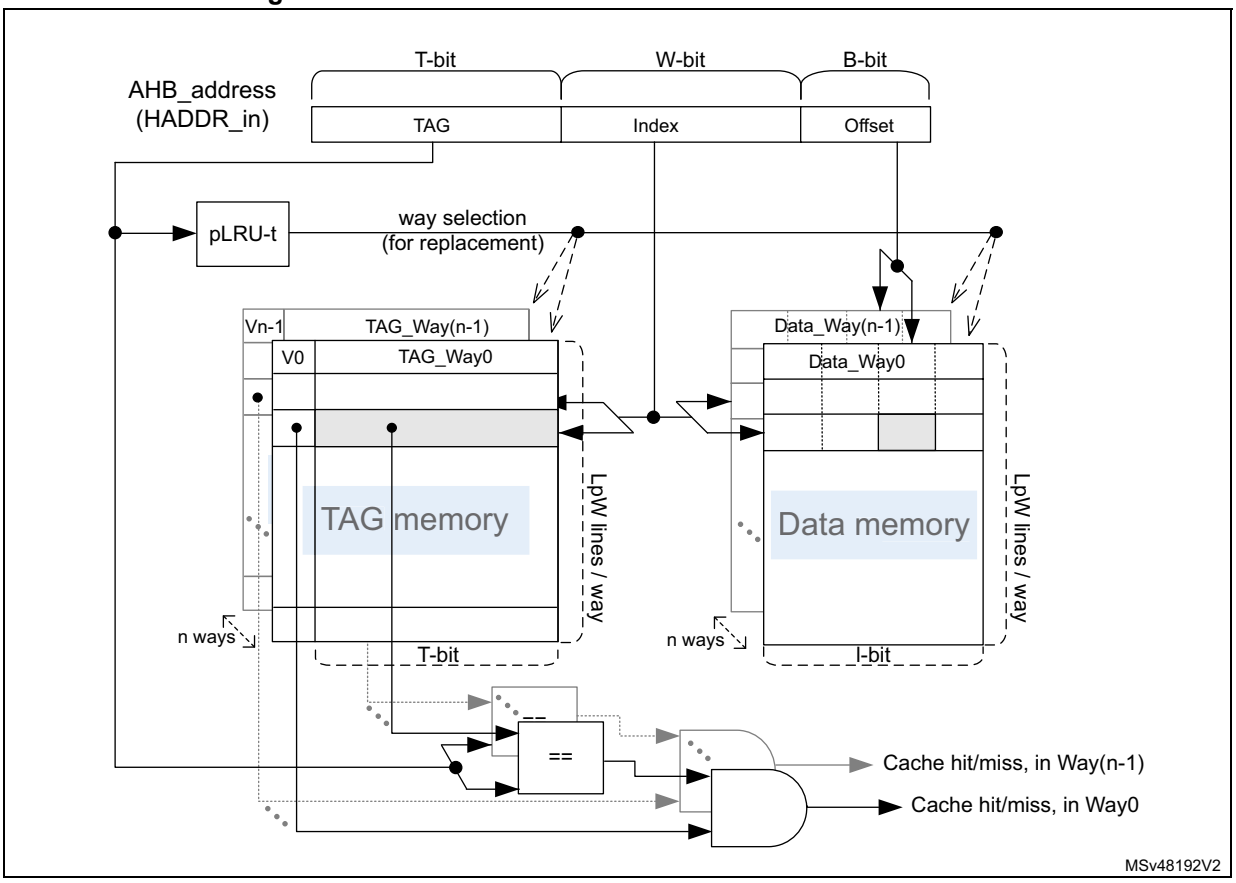
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 46* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 73. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 46. ICACHE TAG and data memories functional view



17.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 74. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 17.4.8*): only one way (the unique one) is possible for any data refill.

17.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 47*).

The ICACHE is, by default, disabled at boot.

17.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 17.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 75*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 75. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 17.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 17.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 76 summarizes programmable configurations of various memories.

Table 76. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

17.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 47 describes the matching and the output address generation.

Figure 47. ICACHE remapping address mechanism

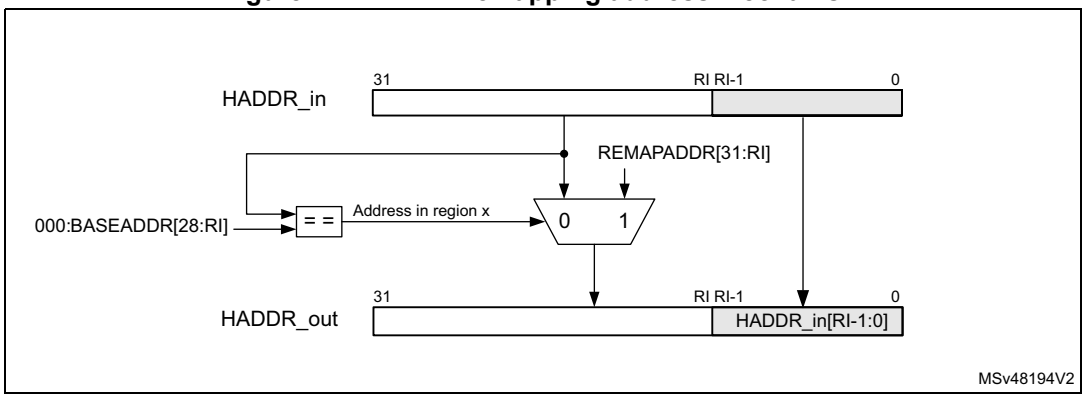


Table 77 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 77. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 77*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

17.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.

- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

17.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

17.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

17.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

17.4.12 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 17.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

17.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

17.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 46*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

17.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 78. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the master1 or master2 port back to the execution port.

17.7 ICACHE registers

17.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

17.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

17.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

17.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
													w	w	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

17.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

17.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

17.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 17.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 17.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

17.7.8 ICACHE hardware configuration register (ICACHE_HWCFGR)

Address offset: 0x3F0

Reset value: 0x00A4 1641

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC	Res.	Res.	Res.	GSWIDTH[1:0]		Res.	SMASTER[1:0]		FMASTER[1:0]		Res.	Res.	REGION[2:0]		
r				r	r		r	r	r	r			r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RANGE[3:0]				Res.	LWIDTH[1:0]		Res.	Res.	SIZE[2:0]			Res.	Res.	WAYS[1:0]	
r	r	r	r		r	r			r	r	r			r	r

Bit 31 **ECC**: error detection and correction support
0: no ECC hardware support
1: ECC hardware support

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:26 **GSWIDTH[1:0]**: data size of the AHB slave interface
00: 32-bit
01: 64-bit
10: 128-bit
11: reserved

Bit 25 Reserved, must be kept at reset value.

Bits 24:23 **SMASTER[1:0]**: data size of the second AHB master interface (master2)
00: no second master (single master cache)
01: 32-bit
10: 64-bit
11: reserved

Bits 22:21 **FMASTER[1:0]**: data size of the first AHB master interface (master1)
00: reserved
01: 32-bit
10: 64-bit
11: 128-bit

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **REGION[2:0]**: number of regions that can be remapped
000: no region
001: 1 region
010: 2 regions
011: 3 regions
100: 4 regions
others: reserved

Bits 15:12 **RANGE[3:0]**: range granularity of remapped memory regions
0000: reserved
0001: 2 Mbytes
0010: 4 Mbytes
0011: 8 Mbytes
0100: 16 Mbytes
0101: 32 Mbytes
0110: 64 Mbytes
0111: 128 Mbytes
others: reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **LWIDTH[1:0]**: cache line width
00: reserved
01: 64-bit (8 bytes per line)
10: 128-bit (16 bytes per line)
11: 256-bit (32 bytes per line)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:4 **SIZE[2:0]**: cache size
000: reserved
001: 2-Kbyte
010: 4-Kbyte
011: 8-Kbyte
100: 16-Kbyte
101: 32-Kbyte
others: reserved

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **WAYS[1:0]**: cache associativity, number of ways
00: reserved
01: 2-way set associative
10: 4-way set associative
11: reserved

17.7.9 ICACHE version register (ICACHE_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: ICACHE major revision

Bits 3:0 **MINREV[3:0]**: ICACHE minor revision

17.7.10 ICACHE identification register (ICACHE_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0081

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: ICACHE identification

17.7.11 ICACHE size identification register (ICACHE_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: ICACHE address space size identification

17.7.12 ICACHE register map

Table 79. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x030-0x3EC	Reserved	Reserved																															
0x3F0	ICACHE_HWCFGR	ECC	Res.	Res.	Res.	GSWIDTH [1:0]		Res.	SMASTER [1:0]		FMASTER [1:0]		Res.	Res.	REGION [2:0]		RANGE[3:0]		Res.	LWIDTH [1:0]		Res.	Res.	SIZE[2:0]			Res.	Res.	WAYS [1:0]			
	Reset value	0x00A4 1641																															
0x3F4	ICACHE_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]			MINREV[3:0]					
	Reset value	0x0000 0012																															
0x3F8	ICACHE_IPIDR	ID[31:0]																															
	Reset value	0x0010 0081																															

Table 79. ICACHE register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3FC	ICACHE_SIDR	SID[31:0]																															
	Reset value	0xA3C5 DD01																															

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

18 Data cache (DCACHE)

18.1 DCACHE introduction

The data cache (DCACHE) is introduced on S-AHB system bus of the Cortex®-M33 processor to improve the performance of data traffic to/from external memories.

Some specific features, like hit-under-miss and critical-word-first refill policy, optimize performance on external memories data accesses.

18.2 DCACHE main features

The main features of DCACHE are described below:

- Bus interface
  - One 32-bit AHB slave port, the system port (input from Cortex®-M33 S-AHB system interface)
  - One 32-bit AHB master port (output to main AHB bus matrix)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for access to DCACHE registers)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimized cache line refill thanks to WRAP bursts of the size of the cache line (such as WRAP4 for 128-bit cache line)
  - 2-way set-associative
  - Supports both write-back and write-through policies (selectable with AHB bufferable attribute)
  - Read and write-back always allocate
  - Write-through always non-allocate (write-around)
  - Supports byte, half-word, and word writes
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy for read transactions, minimizing processor stalls
  - Possibility to configure burst type of all AHB memory transactions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The DCACHE implements four performance counters:
  - Two hit-monitor counters (32-bit): number of read hits, number of write hits
  - Two miss-monitor counters (16-bit): number of read misses, number of write misses

- Error management
  - Possibility to detect error for master port request initiated by DCACHE itself (a cache line written back into main memory, because of an eviction or a clean operation), to flag this error, and optionally to raise an interrupt
- TrustZone security support
- Maintenance operations
  - Cache invalidate: full cache invalidation, fast command, non-interruptible
  - Cache invalidate range: invalidates cache lines whose address belongs to defined range, background task, interruptible
  - Cache clean range: cleans cache lines (if dirty bit = 1, write back line, then clear dirty bit) whose address belongs to defined range, background task, interruptible
  - Cache clean and invalidate range: cleans and invalidates cache lines (if dirty bit = 1, write back line, then invalidate it) whose address belongs to defined range, background task, interruptible

18.3 DCACHE implementation

The DCACHE is placed on Cortex-M33 S-AHB bus, and caches the external memory regions (OCTOSPI, FMC) and DDR.

Indeed, by placing a busmatrix demultiplexing node in front of the DCACHE, S-AHB bus memory requests addressing SRAM region or peripherals region (respectively in ranges [0x2000 0000:0x3FFF FFFF] and [0x4000 0000:0x5FFF FFFF]) are routed directly to the Cortex-M33 AHB busmatrix and DCACHE is bypassed.

Table 80. DCACHE features

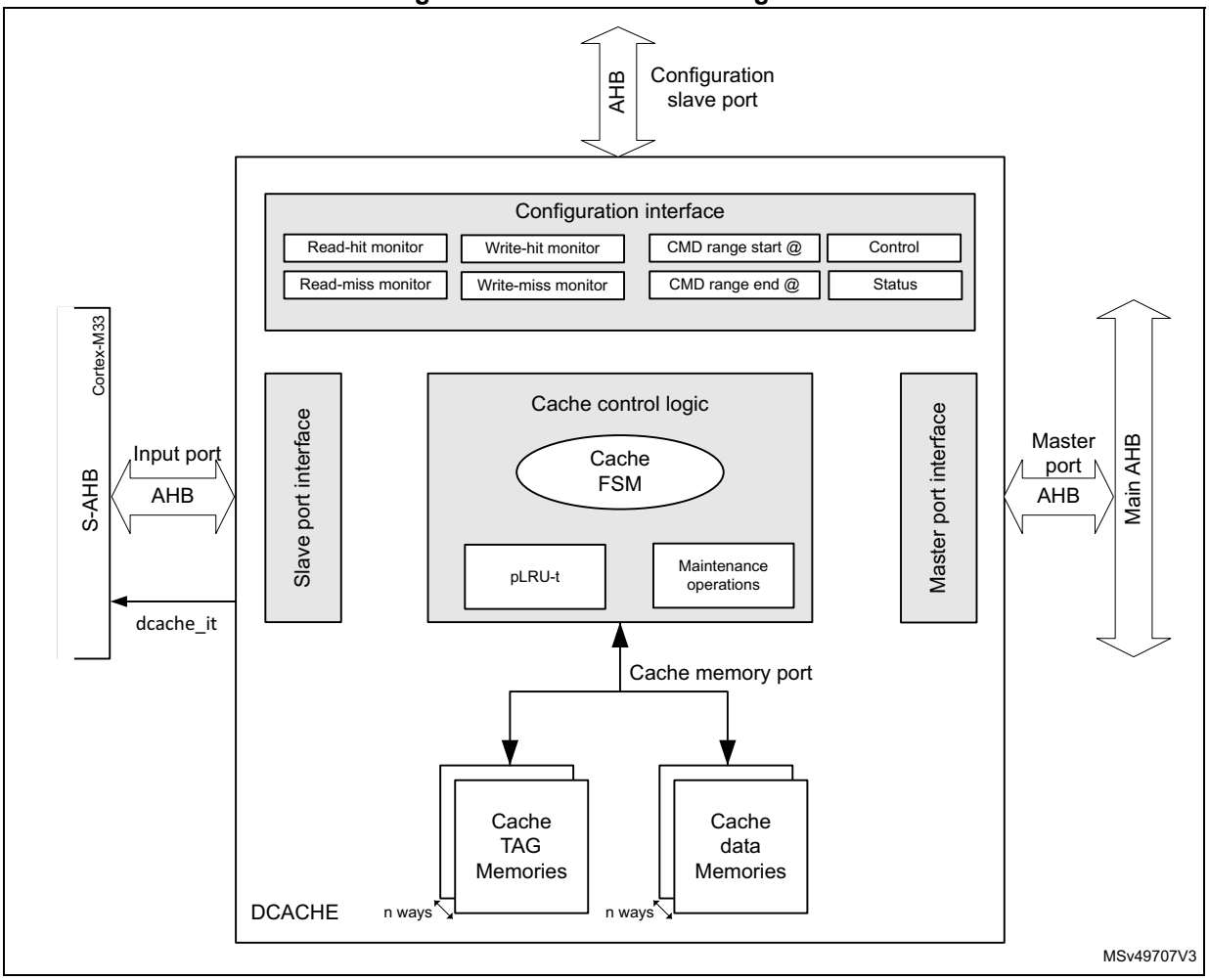
Features	DCACHE
Number of ways	2
Cache size	16 Kbytes
Cache line width	32 bytes
Data size of AHB Master interface	32 bits

18.4 DCACHE functional description

The purpose of the data cache is to cache external memory data loads and stores, coming from the processor. These accesses include the instruction fetches that may occur at an external memory address. The DCACHE manages both read and write transactions.

18.4.1 DCACHE block diagram

Figure 48. DCACHE block diagram



18.4.2 DCACHE reset and clocks

The DCACHE is clocked on the Cortex®-M33 S-AHB bus clock.

When the DCACHE reset signal is released, a cache invalidate procedure is automatically launched, making the DCACHE busy (DCACHE_SR = 0x0000 0001).

When this procedure is finished:

- The DCACHE is invalidated: cold cache, with all cache line valid, dirty, and privilege bits = 0 (DCACHE must be filled up)
- DCACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- The DCACHE is disabled: the EN bit in DCACHE_CR holds its reset state (= 0).

Note: When disabled, the DCACHE is bypassed: slave input requests are just forwarded to the master port.

18.4.3 DCACHE TAG memory

The DCACHE TAG memory contains:

- address tags that indicate which data are contained in the cache data memories
- validity bits
- dirty bits
- privilege bits

There is one valid bit, one dirty bit, and one privilege bit per cache line (per way).

The valid bit enables/disables access to the data cache line: if the line is not valid, the data access (read or write) is performed in the main memory.

The valid bit is set when the cache line is written (refilled by either a read miss or a write-back miss).

Valid bits are reset in any of the below cases:

- after the DCACHE reset is released
- when the cache is disabled, by setting EN = 0 in DCACHE_CR (by software)
- when executing one of the DCACHE invalidate commands, setting by software CACHEINV = 0, or CACHECMD = 0b010 or 0b011 in DCACHE_CR (see *Section 18.4.8*).

The dirty bit indicates that the cache line has up-to-date values with respect to the main memory content (the cache has last right value, the main memory is not up to date).

The dirty bit is set when the cache line is written by a slave port write transaction (only in case of an access with write-back attribute).

Dirty bits are reset in any of the below cases:

- after the DCACHE reset is released
- when a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data, and dirty bit = 1)
- when the cache invalidation is performed
- when executing one of the DCACHE clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b001 or 0b011 in DCACHE_CR (see *Section 18.4.8*).

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AHB privileged attribute at the input slave port, for the first access to this line (it is written only during the line refill, on read miss or write-back miss).

The privilege bit holds the same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

Privilege bits are reset when the cache is invalidated, and after the DCACHE reset is released.

When a cacheable transaction is received at input slave port, its AHB address (HADDR_in) is split into the following fields (see the table below for B and W values):

- HADDR_in[B-1:0]: address byte offset, indicates which byte to select within a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) within the DCACHE

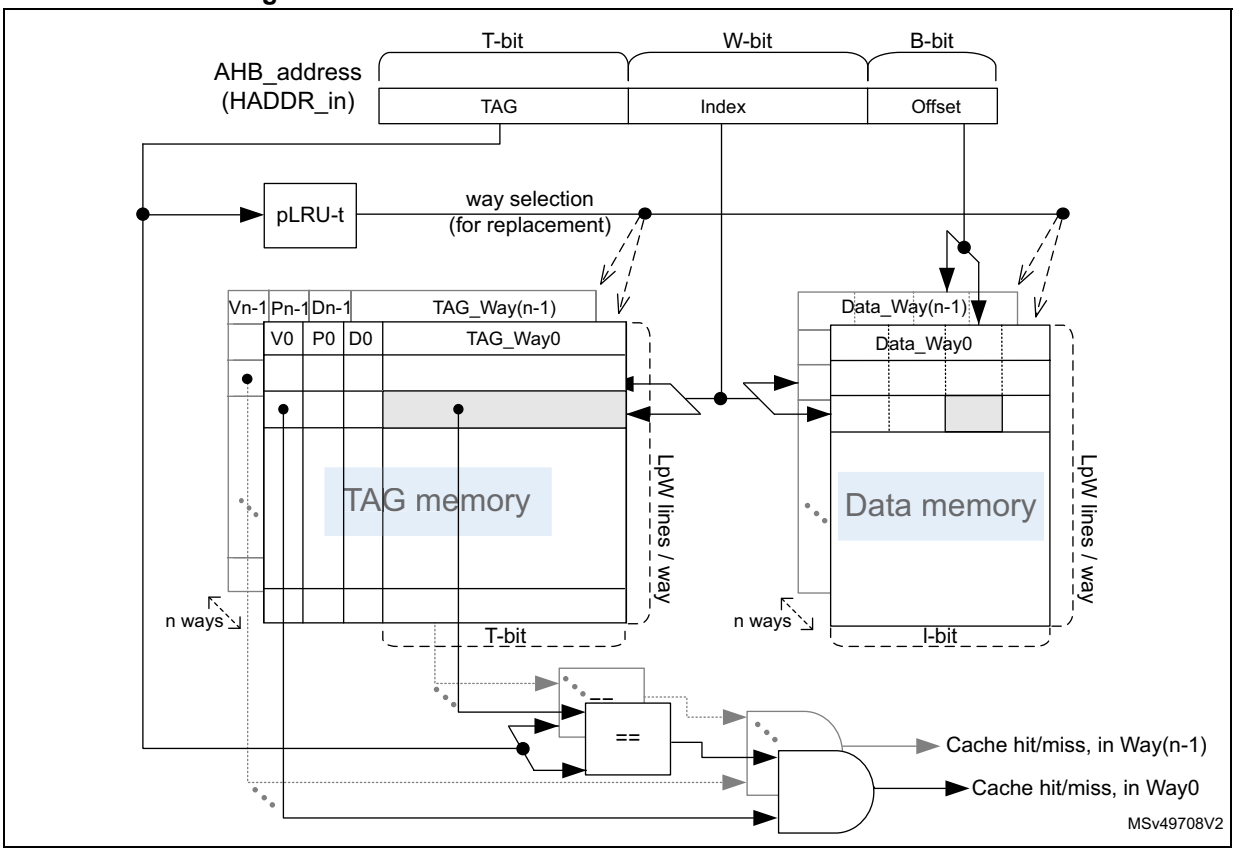
Table 81 gives DCACHE main parameters for TAG memory dimensioning.

Figure 49 shows the functional view of TAG and data memories, for an n-way set associative DCACHE.

Table 81. TAG memory dimensioning parameters

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	4 Kbytes = 4096 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines (per way)	LpW = s / (n x L) lines/way	128 lines/way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	7-bit
TAG address size	T = (32 - W - B) bit	21-bit

Figure 49. DCACHE TAG and data memories functional view



18.4.4 DCACHE enable

To activate the DCACHE functioning, the EN bit must be set in DCACHE_CR control register.

When DCACHE is disabled, it is bypassed and all transactions are copied from slave port to master port in the same clock cycle, and no comparison is performed with TAG address.

DCACHE is by default disabled at boot.

18.4.5 Cacheable and noncacheable traffic

DCACHE is developed for a Cortex®-M33 core and caches the memory regions addressed by the AHB bus connected to it.

In addition, the AHB bus traffic to the memory regions can be cacheable or noncacheable. An incoming memory request to DCACHE is defined as cacheable according to its AHB transaction memory lookup attribute.

In case of write transaction, the DCACHE write policy is defined as write-through or write-back according to its AHB transaction memory bufferable attribute (see *Table 82*).

These AHB attributes depend on the memory protection unit (MPU) programming for the addressed region.

Table 82. DCACHE cacheability for AHB transaction

AHB lookup attribute	AHB bufferable attribute	Cacheability
0	x	Read and write: non cacheable
1	0	Read: cacheable Write: (cacheable) write-through
1	1	Read: cacheable Write: (cacheable) write-back

In case of noncacheable access, the DCACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In case of cacheable access, the DCACHE behaves as explained in *Section 18.4.6*.

Cacheable memory regions are defined and programmed by the user in the MPU, responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

18.4.6 Cacheable accesses

When the DCACHE receives a cacheable transaction from Cortex®-M33 on its slave port, the DCACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

For **read** transaction, there are three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache read hit**, the data is read from cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache read miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first refill policy insures minimum wait cycles for the processor, since read data can be provided while cache is still performing cache line refill (associated latency is the latency of fetching one word from main memory).

  The kind of burst generated on the DCACHE master bus depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated from the slave input (missing) request to the master output refill request.
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache read hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can provide the requested read data as soon as the data is available at its master interface, thus avoiding a miss (with data fetch from main memory).

For **write-back** transaction (write transaction, with write-back bufferable attribute), there are three alternatives as well:

- The address is present inside the TAG memory, cache line is valid: **cache write-back hit**, the data is written in cache.
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-back miss**.

  First, a line allocation is performed by reading the entire cache line data from main memory. The kind of burst generated on the DCACHE master bus for this line refill depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words), and the AHB transaction attributes are propagated from the slave port initial request.

  Once the refilled line is written in the DCACHE, the initial data provided on the slave port is written in this DCACHE line (it overwrites the data part of the cache line that was refilled just before).
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache write-back hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can write incoming data directly inside the refilled line, thus avoiding a miss (with refill from main memory).

For **write-through** transaction (write transaction, with write-through bufferable attribute), only two alternatives exist:

- The address is present inside the TAG memory, cache line is valid: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-through miss**, the data incoming at slave port is written only in main memory (unlike the write-back miss, there is no line allocation and data written in cache).

In case of cache refill (due to cache miss), the DCACHE selects which cache line is written with the refill data: as a 2-way set associative cache, one line among two can be used (line pointed by the address index, in each of the two ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be evicted first:

- If the dirty tag of this line equals 0 (clean data), the line is simply invalidated.
- If it equals 1 (dirty data), the line must be written back in the main memory.

  The DCACHE generates a burst write transaction on its master port, with burst type set to INCRw (w being the cache line width, in words), and with AHB memory transaction attribute signals set as below:
  - data (not instruction)
  - privileged = TAG privilege bit
  - write-back (even if it does not care)
  - normal memory
  - cacheable
  - allocate (even if it does not care)

– nonshareable

These AHB attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted by the DCACHE) because the evicting transaction has no relation with the initial missing transaction. Setting of the AHB attributes is fixed, except for the privileged bit that is copied from the TAG privilege bit of the evicted line.

18.4.7 DCACHE security

The DCACHE implements an Armv8-M TrustZone.

DCACHE configuration registers are protected at system level.

18.4.8 DCACHE maintenance

The DCACHE features several maintenance operations that the software can programmed in DCACHE_CR control register:

- **Full invalidate**: invalidates the whole cache, non interruptible task.

  The software can invalidate the whole DCACHE content by programming CACHEINV in DCACHE_CR.

  When CACHEINV = 1, the DCACHE control logic sets BUSYF flag in DCACHE_SR status register, and performs the operation of cache invalidation, resetting each TAG valid bit to 0 (one valid bit per cache line). Each dirty and privilege bits are also reset to 0 during cache invalidation to prevent unknown values at next cache line validation. CACHEINV is automatically cleared.

  Once the full invalidate operation is finished, the DCACHE automatically clears BUSYF flag, and sets BSYENDF in DCACHE_SR.

  If enabled on this flag condition (BSYENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised. Then, the (empty) cache is available again.

  This full invalidate operation is not interruptible, meaning that the cache does not treat any cacheable request while BUSYF = 1. However, non-cacheable traffic is treated (since the request address is not compared to TAG ones), the DCACHE being bypassed in the same clock cycle (same behavior as when the DCACHE is disabled).

- **Invalidate range**: invalidates a certain range of addresses in the cache, background task (interruptible).

  The software can invalidate a given data region in the DCACHE by programming STARTCMD = 1 and CACHECMD = 0b010 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), the

corresponding cache line is invalidated (line TAG bits cleared, valid bit = dirty bit = privilege bit = 0).

When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the invalidate range operation. STARTCMD is also automatically cleared.

Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

During this invalidate range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the invalidation process. The TAG memory is accessed for invalidate range operation only if not already accessed by an external cache request. This implies that invalidate range execution is usually not performed in one go, but can be interrupted.

- **Clean range**: cleans a certain range of addresses in the cache, background task (interruptible).

  Cleaning a cache line means making sure that the main memory content is up-to-date with the data, which may have been modified in the cache. The clean operation consists in performing the write-back in the main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

  The software can clean a given data region in DCACHE by programming STARTCMD = 1, and CACHECMD = 0b001 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), and the corresponding line is dirty, this line is cleaned, meaning the whole cache line is written-back in the memory through the DCACHE master port, and its TAG dirty bit is cleared.

  When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the clean range operation. STARTCMD is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

  If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

  During this clean range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the cleaning process. The TAG memory is accessed for clean range operation only if not already accessed by an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

  It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until clean range is completed. For that, the software can take advantage of BUSYCMDF flag in DCACHE_SR, and can poll this flag to prevent any spurious access to the area being cleaned.

Alternatively it can also rely on the command end flag (CMDENDF) or on the DCACHE interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates a certain range of addresses in the cache, background task (interruptible).

  This operation cleans the “dirty” cache lines that belong to the operation address range (the same as the clean range operation), and also invalidates all the (valid) cache lines that belong to this address range (whether they are dirty or not).

  The software can launch this clean and invalidate range operation, by programming STARTCMD = 1, and CACHECMD = 0b011 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  This sets and clears the same flags, and potentially the same interrupt as invalidate range or clean range operations.

18.4.9 DCACHE performance monitoring

The DCACHE provides the following monitors for performance analysis:

- The two 32-bit read-hit and write-hit monitors count the AHB transactions at the DCACHE input (slave port) that do not generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is present in the TAG memory, or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 18.4.6*).
- The two 16-bit read-miss and write-miss monitors count the AHB transactions at the DCACHE input (slave port) that generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is not present neither in the TAG memory, nor in the refill buffer.

Upon reaching their maximum values, the monitors do not wrap over.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through R(W)HITMEN in DCACHE_CR.
- Reset the read (write) hit monitor, by setting R(W)HITMRST in DCACHE_CR.
- Enable/stop the read (write) miss monitor, through R(W)MISSMEN in DCACHE_CR.
- Reset the read (write) miss monitor, by setting R(W)MISSMRST in DCACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

18.4.10 DCACHE boot

The DCACHE is disabled (EN = 0 in DCACHE_CR) at boot.

Once the boot is finished, the DCACHE can be enabled (software setting EN = 1 in DCACHE_CR).

18.5 DCACHE low-power modes

At product level, the DCACHE reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even much higher, since the cached main memories are external.

18.6 DCACHE error management and interrupts

A transaction initiated on the DCACHE master port may return an error (a write attempt into a read-only memory, for instance). If the master port request was propagated from a slave port request, the error is propagated back to the slave port. If ever the master port request is initiated by the DCACHE itself (a cache line is written back into the main memory because of an eviction or a clean operation), the DCACHE receives this functional error and flags it internally by setting the ERRF flag in DCACHE_SR.

In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in DCACHE_IER).

Another case of potential interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, the DCACHE sets BSYENDF flag in DCACHE_SR.

An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in DCACHE_IER).

Last case is at the end of invalidate and/or clean range operations: when the command busy state is finished, the DCACHE sets CMDENDF flag in DCACHE_SR.

An interrupt is also generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in DCACHE_IER).

All DCACHE interrupt sources raise the same and unique interrupt signal, dcache_it, and then use the same interrupt vector.

Table 83. DCACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
DCACHE	Functional error	ERRF in DCACHE_SR	ERRIE in DCACHE_IER	Set CERRF to 1 in DCACHE_FCR
	End of busy state (full invalidate finished)	BSYENDF in DCACHE_SR	BSYENDIE in DCACHE_IER	Set CBSYENDF to 1 in DCACHE_FCR
	End of cache operations (address range based)	CMDENDF in DCACHE_SR	CMDENDIE in DCACHE_IER	Set CCMDENDF to 1 in DCACHE_FCR

The DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port.

18.7 DCACHE registers

18.7.1 DCACHE control register (DCACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
				w	rw	rw	rw							w	rw

Bit 31 **HBURST:** output burst type for cache master port read accesses
Can be set by software, only when EN = 0.
Master port write accesses are always done in INCR burst type.
0: WRAP
1: INCR

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **WMISSMRST:** write-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-miss monitor

Bit 22 **WHITMRST:** write-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-hit monitor

Bit 21 **WMISSMEN:** write-miss monitor enable
0: cache write-miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-miss monitor enabled

Bit 20 **WHITMEN:** write-hit monitor enable
0: cache write-hit monitor switched off. Stopping the monitor does not reset it.
1: cache write-hit monitor enabled

Bit 19 **RMISSMRST:** read-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-miss monitor

Bit 18 **RHITMRST:** read-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-hit monitor

Bit 17 **RMISSMEN:** read-miss monitor enable
0: cache read-miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-miss monitor enabled

Bit 16 **RHITMEN:** read-hit monitor enable
0: cache read-hit monitor switched off. Stopping the monitor does not reset it.
1: cache read-hit monitor enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **STARTCMD:** starts maintenance command (maintenance operation defined in CACHECMD).
Can be set by software, only when EN = 1, BUSYCMDF = 0, BUSYF = 0 and CACHECMD = 0b001, 0b010 or 0b011.
Cleared by hardware when the BUSYCMDF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: command operation (cache maintenance) finished
1: start maintenance command (cache maintenance)

Bits 10:8 **CACHECMD[2:0]:** cache command maintenance operation (cleans and/or invalidates an address range)
Can be set and cleared by software, only when no maintenance command is ongoing (BUSYCMDF = 0).
000: no operation
001: clean range
010: invalidate range
011: clean and invalidate range
others: reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CACHEINV:** full cache invalidation
Can be set by software, only when EN = 1.
Cleared by hardware when the BUSYF flag is set (during full cache invalidation operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN:** enable
0: cache disabled
1: cache enabled

18.7.2 DCACHE status register (DCACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDF:** command end flag
Cleared by writing DCACHE_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished

Bit 3 **BUSYCMDF:** command busy flag
0: cache not busy on a CACHECMD command
1: cache busy on a CACHECMD command (clean and/or invalidate an address range)

Bit 2 **ERRF:** cache error flag
Cleared by writing DCACHE_FCR.CERRF = 1.
0: no error
1: an error occurred during the operation (eviction or clean operation write-back error).

Bit 1 **BSYENDF:** full invalidate busy end flag
Cleared by writing DCACHE_FCR.CBSYENDF = 1.
0: cache busy or in idle
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF:** full invalidate busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

18.7.3 DCACHE interrupt enable register (DCACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
											rw		rw	rw	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDIE**: interrupt enable on command end
Set by software to enable an interrupt generation at the end of a cache command (clean and/or invalidate an address range)
0: interrupt disabled on command end
1: interrupt enabled on command end

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRIE:** interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (eviction or clean operation write-back error)
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE:** interrupt enable on busy end
Set by SW to enable an interrupt generation at the end of a cache full invalidate operation.
0: Interrupt disabled on busy end
1: Interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

18.7.4 DCACHE flag clear register (DCACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
											w		w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CCMDENDF:** clear command end flag
Set by software.
0: no effect
1: clears CMDENDF flag in DCACHE_SR

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CERRF:** clear cache error flag
Set by software.
0: no effect
1: clears ERRF flag in DCACHE_SR

Bit 1 **CBSYENDF:** clear full invalidate busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in DCACHE_SR

Bit 0 Reserved, must be kept at reset value.

18.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RHITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RHITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RHITMON[31:0]:** cache read-hit monitor counter

18.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RMISSMON[15:0]:** cache read-miss monitor counter

18.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WHITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WHITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WHITMON[31:0]:** cache write-hit monitor counter

18.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WMISSMON[15:0]:** cache write-miss monitor counter

18.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDSTARTADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSTARTADDR[15:5]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:5 **CMDSTARTADDR[31:5]:** start address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 4:0 Reserved, must be kept at reset value.

18.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDENDADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDENDADDR[15:5]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:5 **CMDENDADDR[31:5]:** end address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 4:0 Reserved, must be kept at reset value.

18.7.11 DCACHE hardware configuration register (DCACHE_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0080 0641

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC	Res.	Res.	Res.	Res.	Res.	Res.	MASTERSIZE [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.
r							r	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LWIDTH[1:0]		Res.	Res.	SIZE[2:0]			Res.	Res.	WAYS[1:0]	
					r	r			r	r	r			r	r

Bit 31 **ECC**: error detection and correction support
0: no ECC hardware support
1: ECC hardware support

Bits 30:25 Reserved, must be kept at reset value.

Bits 24:23 **MASTERSIZE[1:0]:** data size of AHB master interface
00: reserved
01: 32-bit
10: 64-bit
11: reserved

Bits 22:11 Reserved, must be kept at reset value.

Bits 10:9 **LWIDTH[1:0]:** cache line width
00: reserved
01: 64-bit (8 bytes per line)
10: 128-bit (16 bytes per line)
11: 256-bit (32 bytes per line)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:4 **SIZE[2:0]:** cache size
000: reserved
001: 2-Kbyte
010: 4-Kbyte
011: 8-Kbyte
100: 16-Kbyte
others: reserved

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **WAYS[1:0]:** cache associativity, number of ways
00: reserved
01: 2-ways set associative
10: 4-ways set associative
11: reserved

18.7.12 DCACHE version register (DCACHE_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** DCACHE major revision

Bits 3:0 **MINREV[3:0]:** DCACHE minor revision

18.7.13 DCACHE identification register (DCACHE_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0091

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** DCACHE identification

18.7.14 DCACHE size identification register (DCACHE_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** DCACHE address space size identification

18.7.15 DCACHE register map

Table 84. DCACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DCACHE_CR	HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN	Res.	Res.	Res.	Res.	STARTCMD	CACHECMD [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
	Reset value	0								0	0	0	0	0	0	0	0					0	0	0	0							0	0
0x004	DCACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
	Reset value																												0	0	0	0	1
0x008	DCACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
	Reset value																												0		0	0	

Table 84. DCACHE register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	DCACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
	Reset value																												0		0	0	
0x010	DCACHE_ RHMONR	RHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DCACHE_ RMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DCACHE_ WHMONR	WHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DCACHE_ WMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DCACHE_ CMDRSADDRR	CMDSTARTADDR[31:5]																											Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x02C	DCACHE_ CMDREADDRR	CMDENDADDR[31:5]																											Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x030-0x3EC	Reserved	Reserved																															
0x3F0	DCACHE_ HWCFGR	ECC	Res.	Res.	Res.	Res.	Res.	Res.	MASTERSIZE [1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LWIDTH[1:0]	Res.	Res.	SIZE[2:0]			Res.	Res.	WAYS[1:0]	
	Reset value	0x0080 0641																															
0x3F4	DCACHE_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]			MINREV[3:0]				
	Reset value	0x0000 0010																															
0x3F8	DCACHE_IPIDR	ID[31:0]																															
	Reset value	0x0010 0091																															
0x3FC	DCACHE_SIDR	SID[31:0]																															
	Reset value	0xA3C5 DD01																															

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

19 Power control (PWR)

The power control (PWR) provides an overview of the supply architecture of the different power domains, and the control of the supply configuration.

It provides an overview of the power supply supervisor features and provides the control of the V_{DDCPU} and V_{DDCORE} supply domains according to the operating modes. In the whole document:

- The Cortex-A35 is also called CPU1 or MPU.
- The Cortex-M33 is also called CPU2 or MCU.

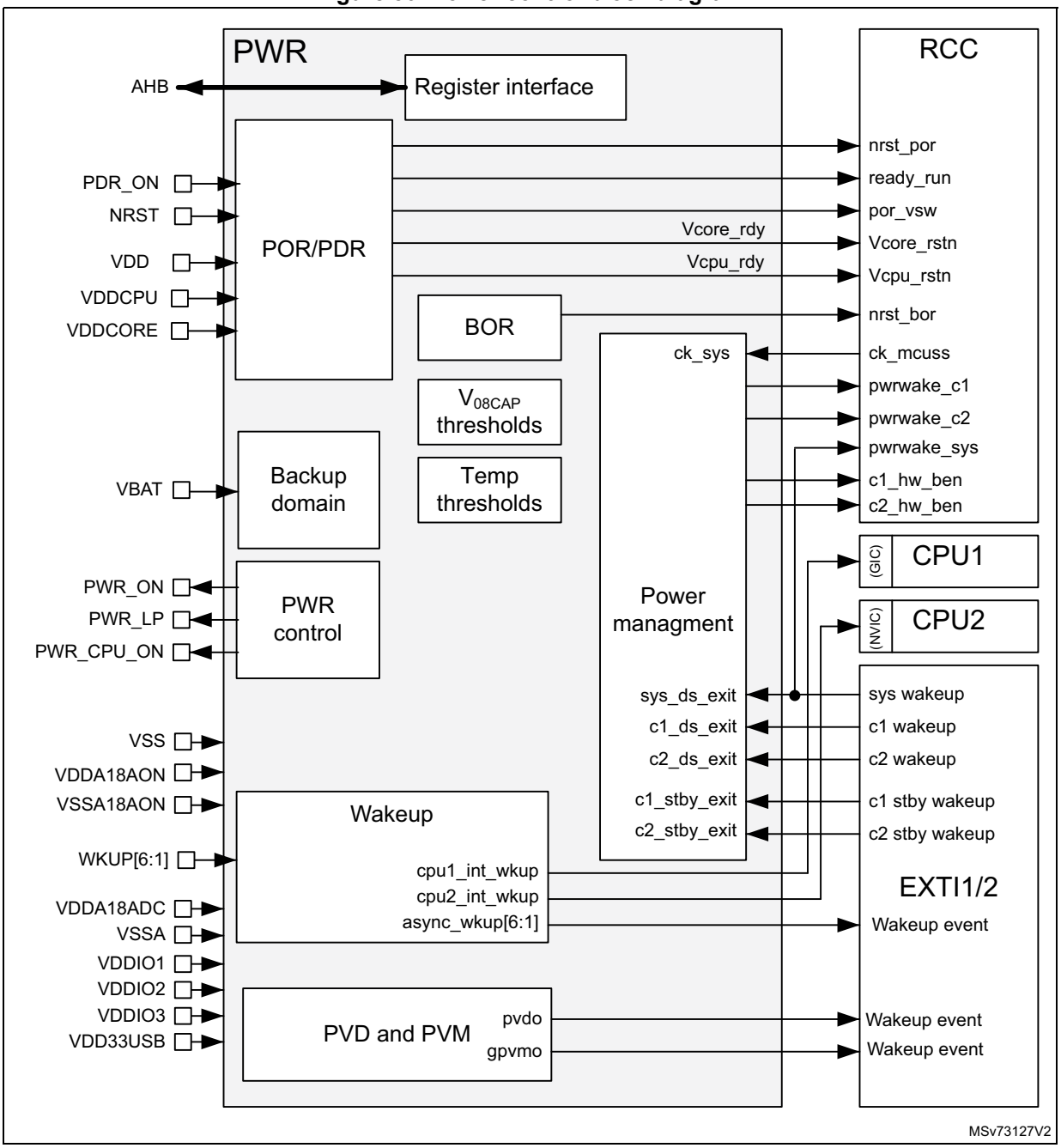
19.1 PWR main features

- System supply external regulation control
  - V_{DDCORE} supply request PWR_ON
  - V_{DDCPU} supply request PWR_CPU_ON
  - V_{DDCORE}/V_{DDCPU} low-power mode control PWR_LP
- Supply regulation
  - 0.8 V backup regulator (external to PWR)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - PVM monitor (V_{DDIO1}, V_{DDOI2}, V_{DDIO3}, $V_{DD33USB}$, $V_{DDA18ADC}$)
  - V_{08CAP} thresholds
  - Temperature thresholds
  - V_{DDCORE} monitor
  - V_{DDCPU} monitor
- Power management
  - Operating modes
  - Low-power modes

19.2 PWR block diagram

The figure below gives a PWR simplified view and the key signals exchanged with the other blocks.

Figure 50. Power control block diagram



19.2.1 PWR pin overview

The table below lists the PWR signals connected to pads or balls.

Table 85. PWR pin overview

Pin name	I/O	Description
VDD	Power	Main I/O and V_{DD} domain supply input
VDDA18AON	Power	Main analog supply input
VBAT	Power	Backup battery supply input
VDDCPU	Power	CPU (Cortex-A35) domain supply
VDDCORE	Power	Core domain supply
VDDA18ADC	Power	ADC analog supply
VDDIO1	Power	Independent I/O supply 1 (PE[5:0])
VDDIO2	Power	Independent I/O supply 2 (PE[15:6])
VDDIO3	Power	Independent I/O supply 3 (PD[11:0])
VDD33USB	Power	USB HS PHY IO 3.3 V supply input
VSS	Power	Main ground
VSSA	Power	Analog ground
PWR_ON	Output	Core supply enable output
PWR_LP	Output	Core/CPU supply low-power Stop mode selection output
PWR_CPU_ON	Output	CPU supply enable output
NRST	Input/output	Application pad reset input/output.
PDR_ON	Input	Used during STMicroelectronics production tests, must always be connected to $V_{DDA18AON}$ in final application.

19.3 PWR power supplies

The system requires supply on VDD, VDDA18AON, VDDCPU and VDDCORE, and provides independent supplies for VDDA18ADC, VBAT, VDD33USB, VDDIO1, VDDIO2, VDDIO3, and VDDQDDR:

- V_{DD} power supply for I/Os
- $V_{DDA18AON}$ power supply for system analog such as reset, power management, oscillators and OTP
- V_{BAT} optional power supply for backup domain when V_{DD} is not present (V_{BAT} mode)
- V_{DDCORE} digital core domain supply, dependent on V_{DD} and $V_{DDA18AON}$ supplies. V_{DD} and $V_{DDA18AON}$ must be present before V_{DDCORE}.
- V_{DDCPU} digital CPU domain supply (Cortex-A35), dependent on V_{DD} and $V_{DDA18AON}$ supplies. V_{DD} and $V_{DDA18AON}$ must be present before V_{DDCPU}.
- $V_{DDA18ADC}$ analog power supply input for ADCs and voltage reference buffers, independent from any other supply.
- V_{REF+} external reference voltage for ADCs, independent from any other supply.
  - supply output when the voltage reference buffer is enabled

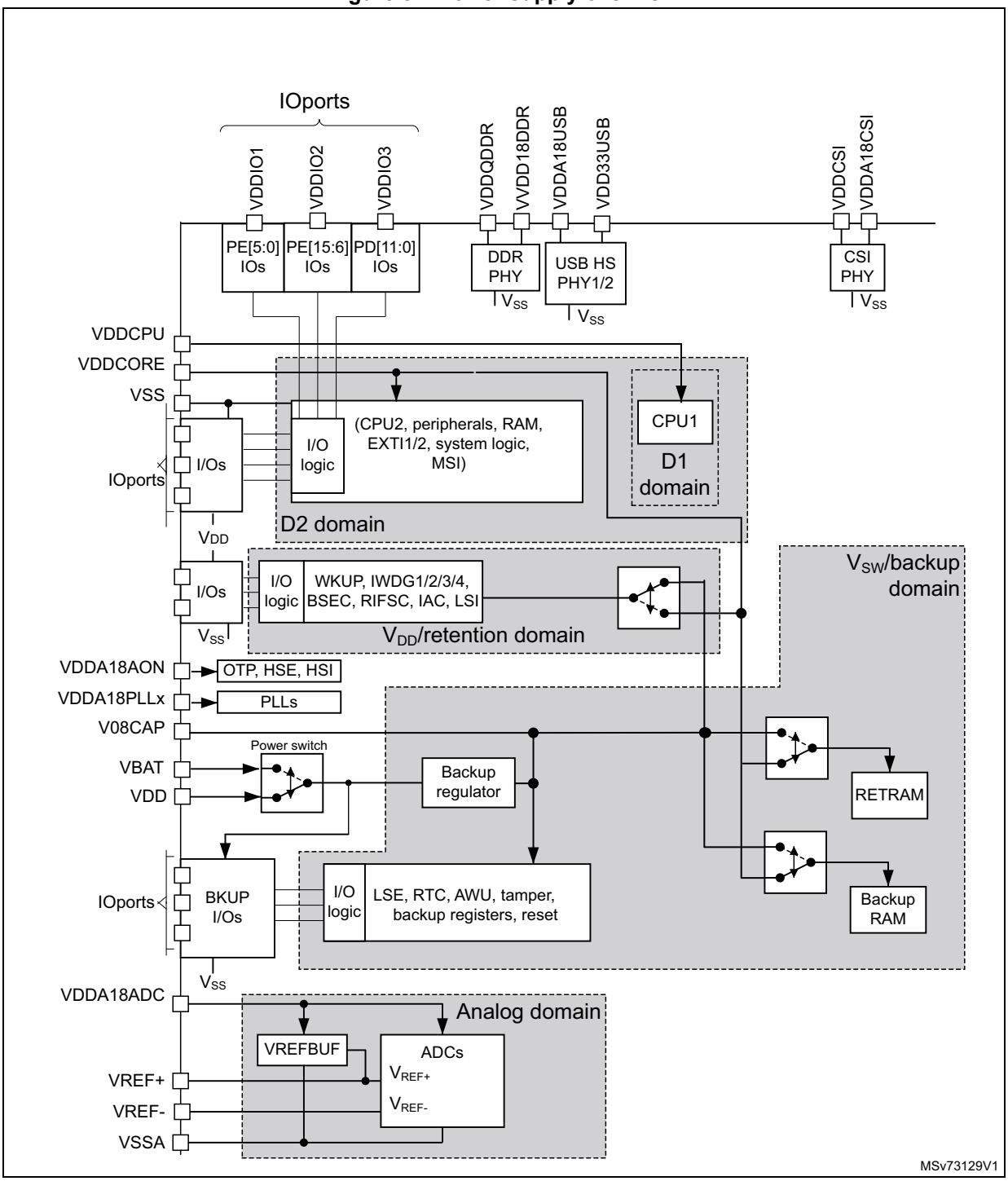
  - independent external reference supply input when the voltage reference buffer is disabled
- V_{SSA} separate analog and reference voltage ground
- $V_{DD33USB}$ supply for USB HS PHY, independent from any other supply
- V_{DDIO1} supply for six I/Os (port E[5:0]), independent from any other supply
- V_{DDIO2} supply for 10 I/Os (port E[15:6]), independent from any other supply
- V_{DDIO3} supply for 12 I/Os (port D[11:0]), independent from any other supply
- $V_{SS33USB}$ separate USB HS PHY ground
- V_{SS} common ground for all supplies except for analog

Note: Depending on the operating power supply range, some peripheral can be used with limited functionality and performance. For more details, refer to section "General operating conditions" in the device datasheet.

An external DDR memory must be supplied from an external regulator which is NOT controlled by the PWR. When the system enters low-power modes it is up to the firmware to control the DDR memory supply directly through the external regulator. When the system enters Standby mode, the PWR_ON pin can be used by the external DDR memory regulator to power down the DDR memory supply.

The figure below gives an overview of the power supply.

Figure 51. Power supply overview



19.3.1 PWR CPU and CORE domains

The V_{DDCPU} CPU domain supply is provided from an external regulator. The V_{DDCPU} supplies the Cortex-A35 platform (CPU1). This domain is called D1 domain.

The V_{DDCORE} core domain supply is provided from an external regulator. The V_{DDCORE} supplies all the digital circuitries (including the Cortex-M33) except for the backup, retention when in Standby. This domain is called system D2 domain.

Following a V_{DD} power-on reset, the external regulators are enabled via PWR_ON or PWR_CPU_ON, and supply the V_{DDCORE} and V_{DDCPU}.

The $V_{DDCPU\ and}$ V_{DDCORE} supplies require special explanation. To save power, these power voltages are changed during the run time of the use cases. The target voltage levels of the V_{DDCPU} can vary for different modes according to the use cases. The modes are the nominal and the overdrive modes. The voltage levels of these supplies can be reduced in low-power modes (low voltage mode) or switched off.

Table 86. V_{DDCPU}/V_{DDCORE} voltage levels (for reference only)

Power rail	Typ value (V)	Description
V_{DDCPU}	0.9	Overdrive mode
	0.8	Nominal mode
	0.67	Low-voltage mode
V_{DDCORE}	0.8	Nominal mode
	0.67	Low-voltage mode

The external regulators can operate in different modes (main, low-power, off) which are used depending on the system operating modes (Run, Stop, LP-Stop, and Standby):

- In Run1/2 modes
  - The regulator supplies full power to the V_{DDCORE} domain. In its main mode, PWR_ON and PWR_LP are set high. In this mode, the PMIC allows the device to change voltage of power rails through the I^2C interface.
  - The external regulator supplying the V_{DDCPU} can be independently controlled using the PWR_CPU_ON pin. When set high, the regulator supplies full power to the V_{DDCPU} domain. When set low, the external regulator supplying V_{DDCPU} domain is switched off (Run2).
- In Stop1/2 modes
  - The regulators supply the V_{DDCORE} and V_{DDCPU} domains to retain the content of registers and internal memories. The external regulators are kept in main mode to allow a fast exit. The low-power mode signaling is disabled with LPDS_D1 in PWR_CPU1R and with LPDS_D2 in PWR_CPU2R. PWR_ON, PWR_CPU_ON, and PWR_LP are kept set high.
  - The external regulator supplying the V_{DDCPU} can be also switched off when the Standby mode is enabled by PDDS_D1 in PWR_CPU1R (Stop2).
- In LP-Stop1/2 modes
  - The regulators supply the V_{DDCORE} and V_{DDCPU} domains to retain the content of registers and internal memories. The external regulators are allowed to enter low-power mode. The low-power mode signaling on PWR_LP is enabled with LPDS_D1 in PWR_CPU1R, and with LPDS_D2 in PWR_CPU2R.

The PWR_CPU_ON is kept set high. PWR_ON is also kept set high, or follows PWR_LP as selected by LPCFG_D2 in PWR_D2CR. PWR_LP is cleared to low.

When the low-voltage Deepsleep mode is enabled by LVDS_D1 in PWR_CPU1R and by LVDS_D2 in PWR_CPU2R, the V_{DDCORE} and V_{DDCPU} supply level can be lowered during LPLV-Stop mode.

  - The external regulator supplying the V_{DDCPU} can be also switched off when Standby mode is enabled by PDDS_D1 in PWR_CPU1R (LP-Stop2).

- In Standby mode
  - PWR_ON, PWR_CPU_ON, and PWR_LP signals are set low. The external regulators are switched off. V_{DDCORE} and V_{DDCPU} domains are powered down. The content of the registers and memories are lost except for the backup and retention domains.

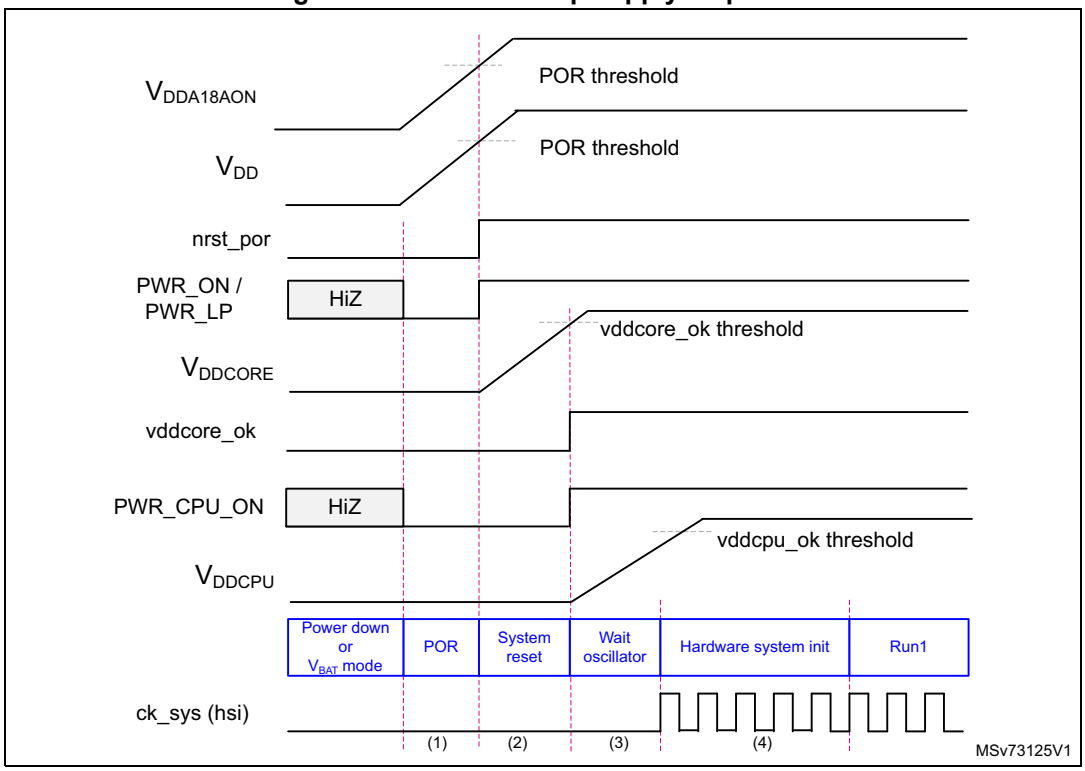
19.3.2 PWR supply system startup sequence

The system startup sequence is detailed below and shown in *Figure 52*:

1. From power-on, the POR monitors V_{DD} and $V_{DDA18AON}$ supplies.

   Once these supplies are above the POR threshold level, the external voltage regulators providing the V_{DDCORE} supply is enabled via the PWR_ON signal.
2. As long as V_{DDCORE} is below the vddcore_ok threshold, the system is kept in reset.
3. Once V_{DDCORE} supply is above the vddcore_ok threshold level, the system is taken out of reset, where after the HSI oscillator is enabled. The external voltage regulator providing the V_{DDCPU} supply is enabled via PWR_CPU_ON signal.
4. Once the oscillator is stable, the hardware initializes the system (option bytes loading), after which Run1 mode is entered if the V_{DDCPU} supply is above the vddcpu_ok threshold.

Figure 52. Device startup supply sequence.

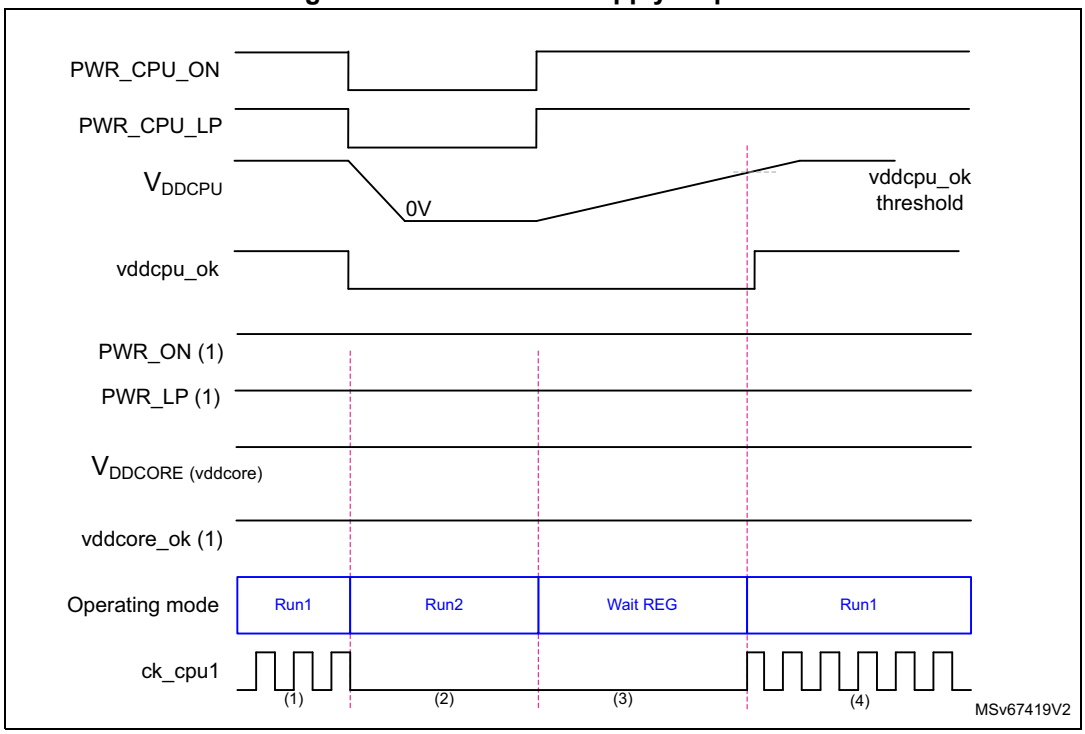


19.3.3 PWR supply Run2 sequence

The system Run2 sequence is detailed below and shown in *Figure 53*.

1. When entering Run2 mode, all CPU1 clocks are stopped.
2. The PWR_CPU_ON signals request the V_{DDCPU} supply to be powered off. If needed, a guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D1 in PWR_D1CR. The PWR_ON and PWR_LP are kept high. The external regulator suppling the V_{DDCORE} remains in Main mode.
3. On a CPU1 wake-up event, the external regulator suppling V_{DDCPU} is requested, via the PWR_CPU_ON signal. As long as V_{DDCPU} is below the vddcpu_ok threshold, the system is kept in Run2 mode. When a wake-up event occurs before the guaranteed minimum PWR_CPU_ON pulse low time, the external regulator is only requested to switch on the V_{DDCPU} supply till after the minimum PWR_CPU_ON pulse low time.
4. Once V_{DDCPU} supply is above the vddcpu_ok threshold level, the CPU1 clocks are enabled and the system enters Run1 mode.

Figure 53. Device Run2 supply sequence.



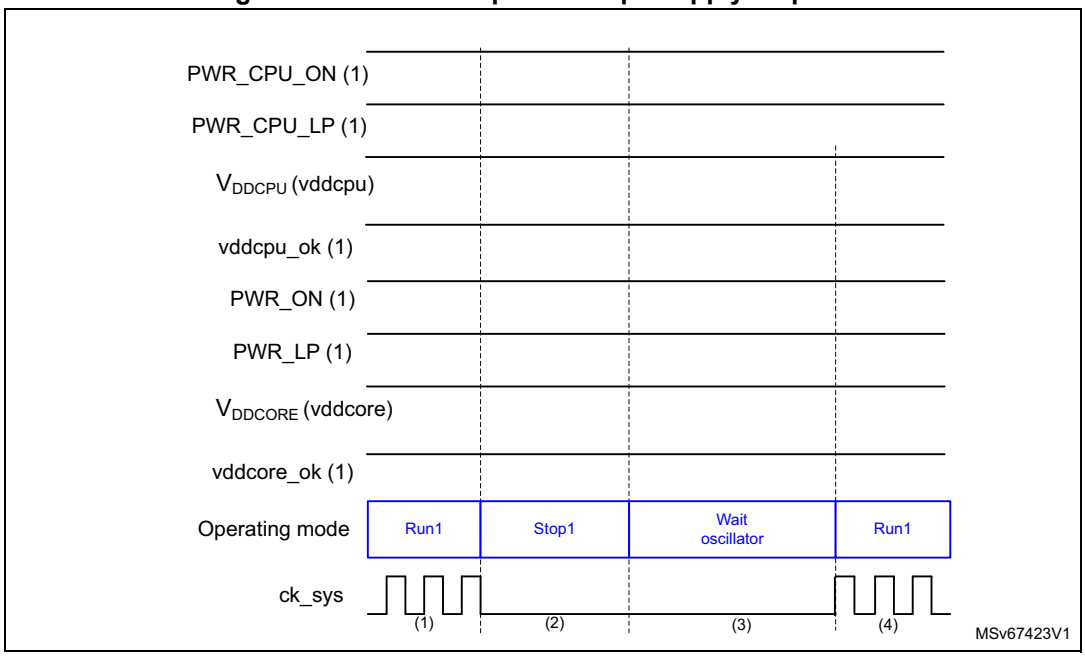
19.3.4 PWR supply Stop sequences

Stop1

The system Stop1 sequence is detailed below and shown in *Figure 54*. Stop1 mode is entered when selected by LPDS_D1 in PWR_CPU1CR, and by LPDS_D2 in PWR_CPU2CR:

1. When entering Stop1 mode, the ck_mcuss (ck_sys) is stopped.
2. In Stop1 mode, the PWR_ON, PWR_CPU_ON, but also PWR_LP are kept high. The external regulators supplying the V_{DDCORE} and V_{DDCPU} remain in Main mode.
3. On a wake-up event, the selected oscillators are enabled.
4. Once the oscillator is stable, the system enters Run1 mode.

Figure 54. Device low-power Stop1 supply sequence

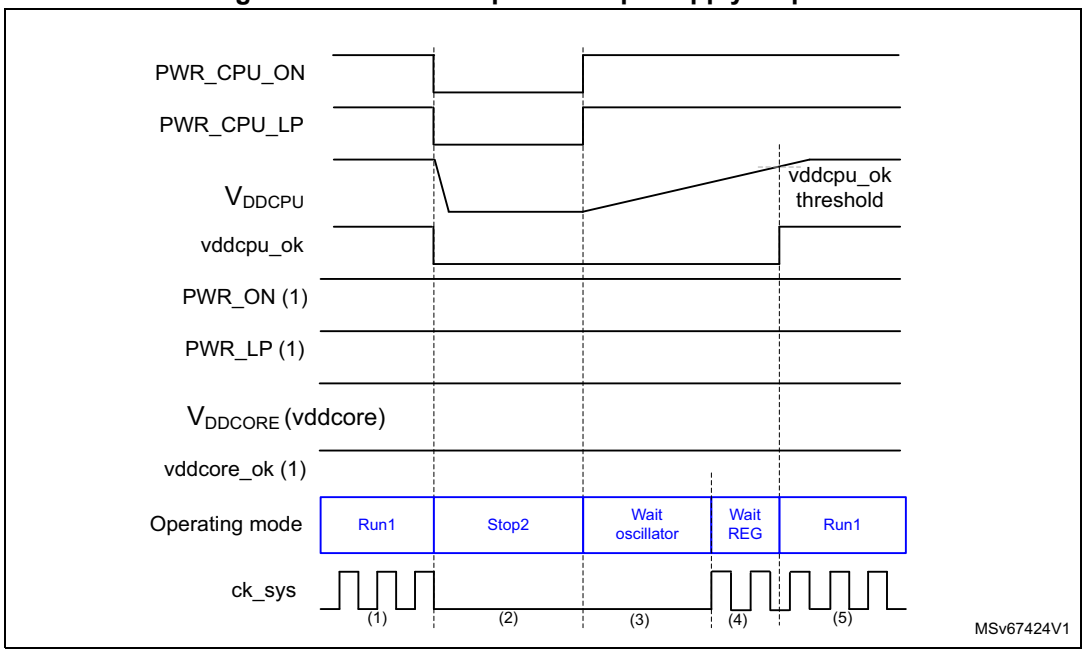


Stop2

The PWR_CPU_ON signal can request the V_{DDCPU} supply to be powered off when D1 Standby mode is selected by PDDS_D1 in PWR_CPU1CR. This mode is called Stop2. The system Stop2 mode sequence is detailed below and shown in *Figure 55*:

1. When entering Stop2 mode, the ck_mcuss (ck_sys) is stopped.
2. The PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off. If needed, a guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D1 in PWR_D1CR. PWR_ON and PWR_LP are kept high. The external regulator suppling the V_{DDCORE} remains in main mode.
3. On a CPU1 wake-up event, the external voltage regulators providing the V_{DDCPU} supply are enabled via the PWR_CPU_ON signal, and the selected oscillators are enabled.
4. Once the oscillator is stable, and as long as the V_{DDCPU} is below the vddcpu_ok threshold, the CORE domain is kept under system reset.
5. Once the V_{DDCPU} is above the vddcpu_ok threshold level, the system enters Run1 mode.

Figure 55. Device low-power Stop2 supply sequence



19.3.5 PWR supply LP/LPV-Stop sequences

LP-Stop1/LPV_Stop1

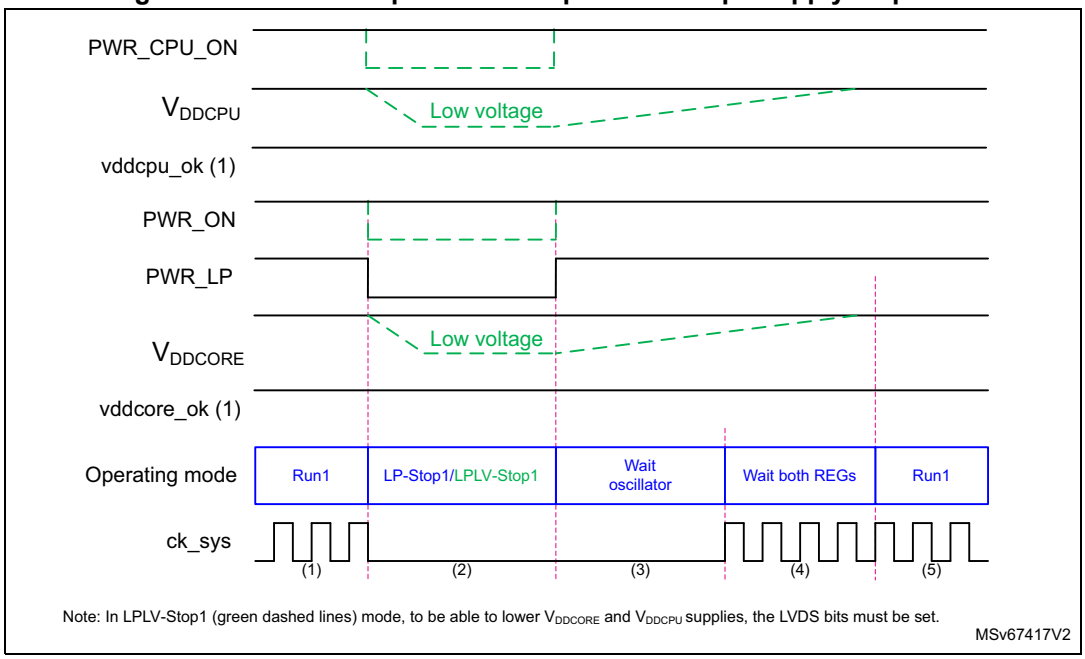
The system LP-Stop1 mode sequence is detailed below and shown in *Figure 56*. LP-Stop1 mode is entered when selected by LPDS_D1 in PWR_CPU1CR and by LPDS_D2 in PWR_CPU2CR:

1. When entering LP-Stop1 mode, the ck_mcuss (ck_sys) is stopped.
2. In LP-Stop1 mode, PWR_CPU_ON and PWR_ON are either kept high (pwr_on) or follow the PWR_LP as selected with LPCFG_D1 in PWR_CPU1CR and LPCFG_D2 in PWR_CPU2CR. The PWR_LP is de-asserted, signaling LP-Stop1. After that, the external regulators can enter low-power mode.

   In LPLV-Stop1 (LVDS bits set), V_{DDCPU} and V_{DDCORE} supplies can be lowered.
3. On a wake-up event, the external regulators are taken out of low-power mode via PWR_LP, and the selected oscillators are enabled.

   In LPLV-Stop1 mode, a delay t_LPLVDLY (defined by LPLVDLY_D2 in PWR_D2CR) is started to wait for V_{DDCPU} and V_{DDCORE} to reach the Run mode operating supply level. After this delay, the oscillators are enabled.
4. Once the oscillator is stable, a delay is timed out to allow the external regulator to switch to Main mode.
5. After the delay timeout, the system enters Run1 mode.

Figure 56. Device low-power LP-Stop1/LPLV-Stop1 supply sequence



LP-Stop2/LPLV-Stop2

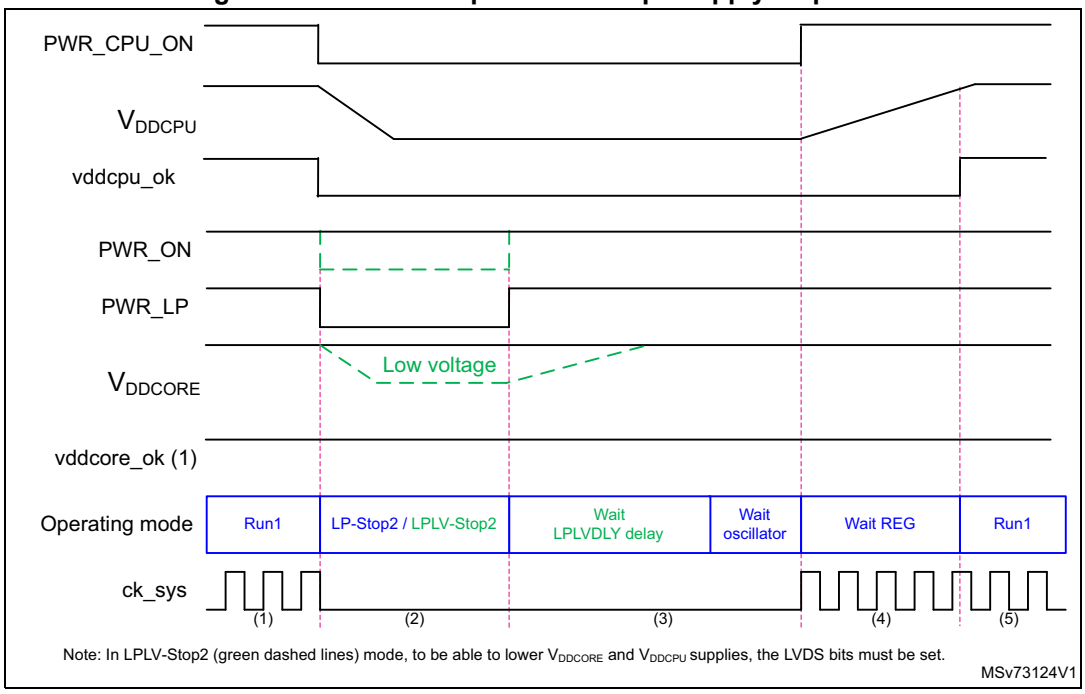
The PWR_CPU_ON signal can request the V_{DDCPU} supply to be powered off when D1 Standby mode is selected by PDDS_D1 in PWR_CPU1CR. This mode is called LP-Stop2. The system LP-Stop2 mode sequence is detailed below and shown in *Figure 57*:

1. When entering LP-Stop2 mode, the ck_mcuss (ck_sys) is stopped.
2. The PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off. If needed, a guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D1 in PWR_D1CR. The PWR_ON is either kept high (pwr_on) or follows the PWR_LP (pwr_onlp) as selected with LPCFG_D2 in PWR_CPU2CR. The PWR_LP is asserted, signaling LP-Stop2. After that, the external regulator can enter low-power mode.
3. On a CPU1 wake-up event, the external regulator suppling V_{DDCORE} is taken out of low-power mode via PWR_LP signal. The selected oscillators are enabled.

   In LPLV-Stop2 mode, a delay t_LPLVDLY (defined by LPLVDLY_D2 in PWR_D2CR) is started to wait for V_{DDCORE} to reach the Run mode operating supply level. After this delay, the oscillators are enabled.
4. Once the oscillator is stable, the external Voltage regulator providing the V_{DDCPU} supply is enabled via the PWR_CPU_ON signal.
5. After the delay timeout, and once V_{DDCPU} is above the vddcpu_ok threshold, the system enters Run1 mode.

Note: If the CPU1 wake-up event is generated from one of the WKUP pins, the PWR_CPU_ON will rise at same time than PWR_LP. The vddcpu_ok delay is then running concurrently than the LPLVDLY and oscillator wait delays, which mean the overall wakeup time to RUN1 is slightly better.

Figure 57. Device low-power LP-Stop2 supply sequence

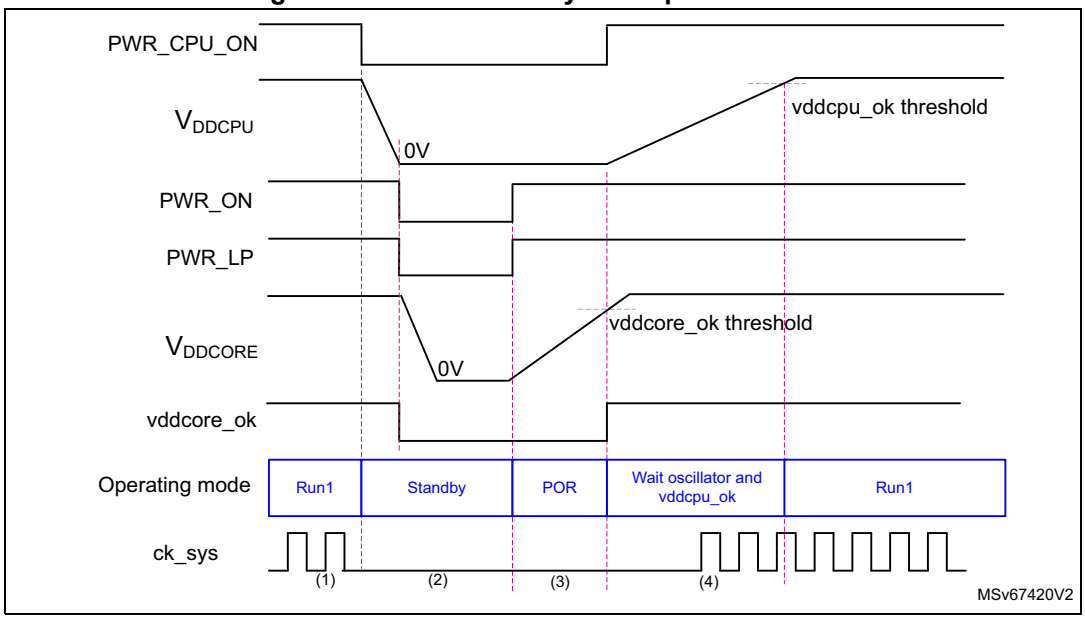


19.3.6 PWR supply Standby sequence

The system Standby mode sequence is detailed below and shown in *Figure 58*:

1. When entering Standby mode, the PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off, after a delay defined by PODH_D2 in PWR_D2CR. If needed, a guaranteed minimum PWR_ON/PWR_CPU_ON pulse low time can be defined by POPL_D1/2 in PWR_D1CR/PWR_D2CR.
2. On a wake-up event, the external regulator suppling the V_{DDCORE} is requested via the PWR_ON signals. As long as V_{DDCORE} is below the vddcore_ok threshold, the system is kept in Standby mode, and V_{DDCPU} is not requested. When a wake-up event occurs before the guaranteed minimum PWR_ON pulse low time, the external regulator is only requested to switch on the V_{DDCORE} supply till after the minimum PWR_ON pulse low time.
3. Once V_{DDCORE} is above the vddcore_ok threshold level, the HSI oscillator is enabled. The PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered on.
4. Once the oscillator is stable, and as long as the V_{DDCPU} supply is requested and below the vddcpu_ok threshold, the system does not enter Run1 mode.

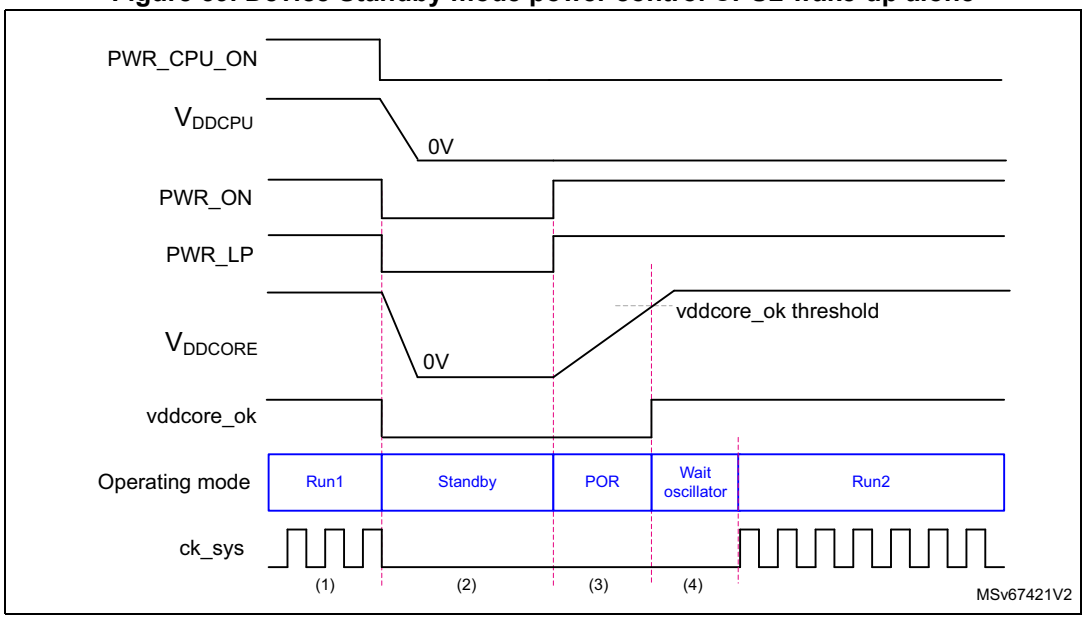
Figure 58. Device Standby mode power control



On a CPU2 wake-up event, and if CPU2 standalone wake-up is allowed by configuration, only the external regulator suppling V_{DDCORE} is switched on via the PWR_ON signals.

The system enters in Run2 mode once the oscillator is stable (as shown in the figure below).

Figure 59. Device Standby mode power control CPU2 wake-up alone

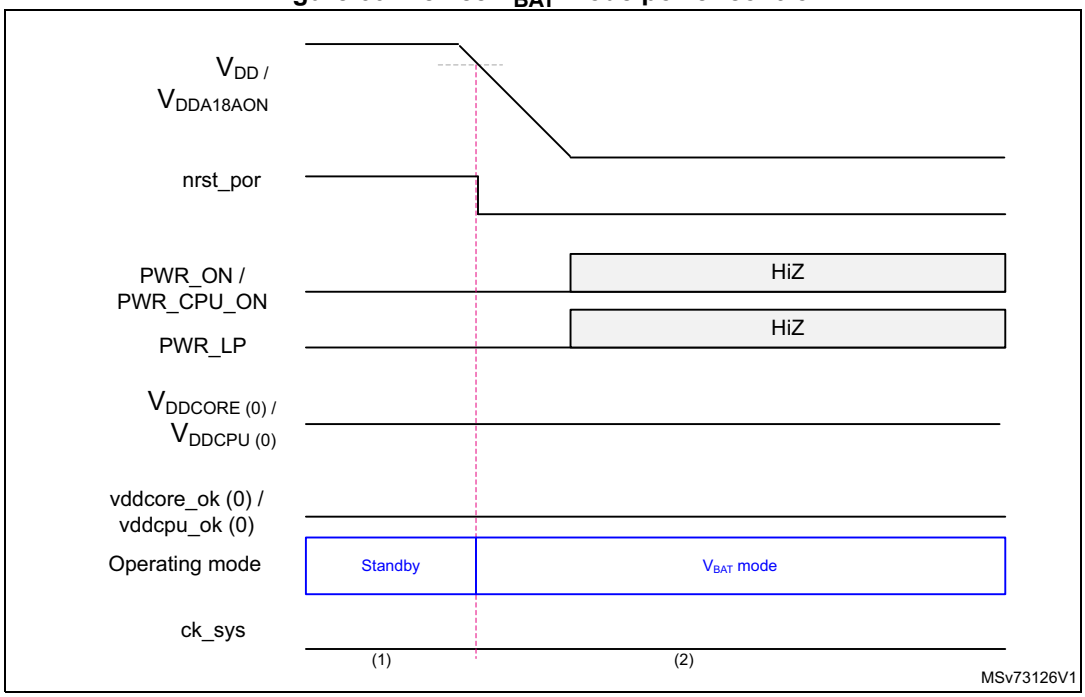


19.3.7 PWR supply V_{BAT} sequence from Standby

The system V_{BAT} mode sequence after Standby is detailed below and shown in *Figure 60*.

1. When in Standby mode, the device can be set in V_{BAT} mode by powering down V_{DD} and $V_{DDA18AON}$ supplies. The PWR_ON and PWR_CPU_ON signals keep requesting V_{DDCORE} and V_{DDCPU} supplies to be powered off.
2. When powering down V_{DD}, the device enters V_{BAT} mode. PWR_ON, PWR_CPU_ON, and PWR_LP signals become HiZ.

Figure 60. Device V_{BAT} mode power control



When V_{BAT} mode is entered from Run1, Stop, or LP-Stop mode, V_{DD}, $V_{DDA18AON}$, V_{DDCORE}, and V_{DDCPU} supplies must all be powered down.

19.3.8 PWR V_{SW} domain

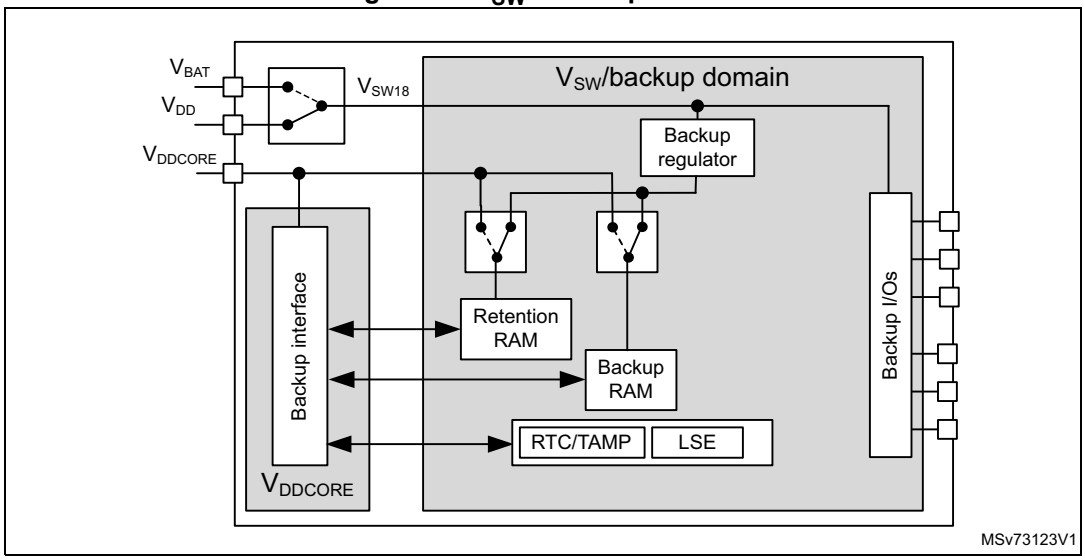
To retain the content of the backup domain (RTC, backup registers, LSE and backup ram) when V_{DD} and $V_{DDA18AON}$ are turned off, the VBAT pin can be connected to an optional standby voltage supplied from a battery or an another source.

Switching to the V_{BAT} supply (V_{BAT} mode) is controlled by the power-down reset (PDR) embedded in the reset block, monitoring the V_{DD} and $V_{DDA18AON}$ supplies.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} and $V_{DDA18AON}$ startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT}

through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

Figure 61. V_{SW} / Backup domain



When the V_{DD} and $V_{DDA18AON}$ supplies are present, the backup domain is supplied from V_{DD}. This allows V_{BAT} power supply battery life time optimization.

If no external backup battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} through a 100 nF external ceramic capacitor.

Whenever V_{DD} and $V_{DDA18AON}$ supplies are present and above the PDR threshold, the backup domain is supplied by V_{DD}. When V_{DD} or $V_{DDA18AON}$ supply is not present and a supply is present on V_{BAT}, the backup domain is supplied by V_{BAT}.

The table below shows the functions available in Backup domain when the core low-power supply is powered off.

Table 87. V_{SW} domain function when V_{DD} and $V_{DDA18AON}$ are presents

Pin name	GPIO	LSE	RTC	TAMPER_OUT(1)	TAMPER_IN(1)
PC3, PC4, PC5, PF6, PF7	-	-	-	-	X(2)
PC13	X	-	X(3)	X	-
PC14, PC15	X	X	-	-	-
PI8	X	-	X(4)	X	X
PZ0, PZ1, PZ3	X	-	-	X	-

1. When configured by the RTC as tamper pin.
2. It is the only function when in V_{SW} domain.
3. See *Table 621: RTC pin PC13 configuration*.
4. See *Table 622: PI8 configuration*.

19.3.9 PWR backup RAM (BKPSRAM)

The backup domain includes an 8-Kbyte backup RAM, accessible in 32-, 16-, or 8-bit data mode.

- In Run, Stop and LP-Stop modes, the BKPSRAM is supplied from the V_{DDCORE} supply.
- In LPLV-Stop, Standby and V_{BAT} modes, the BKPSRAM can be supplied through the backup regulator in the V_{SW} domain.

When the backup supply is enabled by BKPRBSEN in PWR_CR9, the BKPSRAM content is retained even in Standby and/or V_{BAT} mode.

The BKPSRAM can be erased by a tamper event (see *Section 64* for details).

19.3.10 PWR retention RAM (RETRAM)

The backup domain includes a 128-Kbyte retention RAM, accessible in 32-, 16-, or 8-bit data mode.

- In Run, Stop and LP-Stop modes the RETRAM is supplied from the V_{DDCORE} supply.
- In LPLV-Stop, Standby and V_{BAT} modes, the RETRAM is supplied through the backup regulator in the V_{SW} domain.

When the backup supply is enabled by RETRBSEN in PWR_CR10, the RETRAM content is retained even in Standby and/or V_{BAT} mode.

19.3.11 PWR backup domain access

After a system reset, the backup domain (RCC_BDCR, RTC, TAMP and backup registers, backup SRAM) are protected against possible unwanted write accesses. To enable access to the backup domain, DBD in PWR_BDCR must be set.

For more detail on RTC and backup registers, see *Section 63*.

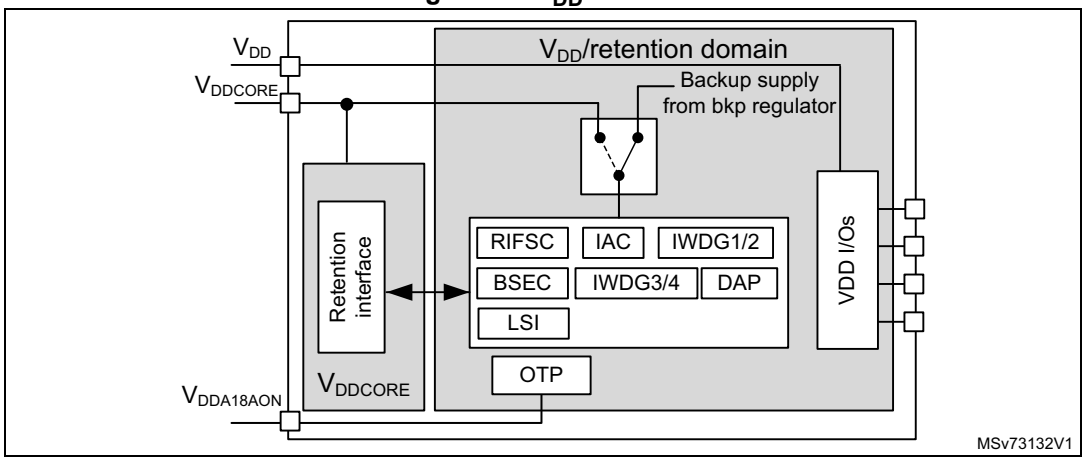
19.3.12 PWR V_{DD} domain

The V_{DD} domain includes boot, security and OTP controller (BSEC), independent watchdog (IWDG), resource isolation framework controller (RIFSC and IAC), and the debug access port (DAP).

- In Run, Stop and LP-Stop modes, the V_{DD} domain is supplied from the V_{DDCORE} supply.
- In LPLV-Stop and Standby modes, the V_{DD} domain is supplied through the backup regulator.

The content of the domain is retained in Standby mode (not powered in V_{BAT} mode).

Figure 62. V_{DD} domain



19.3.13 PWR analog supply

PWR separate analog $V_{DDA18ADC}$ supply

The analog supply domain has separate VDDA18ADC and VSSA pads to allow the supply to be filtered and shielded from noise on the PCB. This improves the ADC conversion accuracy.

- The analog voltage supply input is available on a separate VDDA18ADC pin.
- An isolated supply ground connection is provided on pin VSSA.

PWR analog reference voltage V_{REF}

To allow better accuracy on low-voltage signals, the ADC has a separate reference voltage VREF+ pin. The user can connect a separate external reference voltage on VREF+.
The V_{REF+} supply controls the highest voltage, represented by the full scale value.
The lower voltage reference V_{REF-} must be connected together with V_{SSA}.

When enabled by ENVR in VREFBUF_CSR (see *Section 38.6.1*), the V_{REF+} is provided from the internal voltage reference buffer. The internal voltage reference buffer can also provide a reference voltage to external components through pin (see *Section 38*). The reference voltage can also be generated internally by the ADC cell by setting the SELBG and SELREF bits in ADC_OR (see *Section 36*).

To enable the ADC with an external V_{REF}, the ADC supply must be up and stable before the V_{REF} is enabled. The sequence to monitor the $V_{DD18ADC}$ supply is described in

Section 19.4.6: Peripheral voltage monitoring (PVM). Once done, ENVR and HIZ bit in VREFBUF_CSR register can be used to enable VREF+ pin input (ENVR = 0, HIZ = 1).

19.4 PWR power supply supervision

Power supply level monitoring is available on the following supplies:

- $V_{DD}/V_{DDA18AON}$ via POR/PDR (*Section 19.4.1*), BOR (*Section 19.4.2*), and PVD monitor (*Section 19.4.5*)
  - The POR/PDR monitoring flag is available from PORRSTF in the RCC.
  - The BOR monitoring flag is available from BORRSTF in the RCC.
  - The PVD monitoring flag is available from the PVDO in PWR_CR3. In addition, an interrupt and wake-up can be generated via the EXTI1.
- V_{BAT} via V_{08CAP} threshold (see *Section 19.4.7*)
  - The V_{08CAP} monitoring flags are available from V08CAPH and V08CAPL in PWR_CR2. In addition, an interrupt and wake-up can be generated via the EXTI2.
- V_{DDCORE} via vcore_rdy which keeps V_{DDCORE} domain in reset as long as the level is not OK (see *Section 19.4.3*)
  - The V_{DDCORE} OK monitoring flags are available from SBF in PWR_CPU1CR and PWR_CPU2CR, and from VCORERSTF in the RCC.
- V_{DDCPU} via vcpu_rdy which keeps V_{DDCPU} domain in reset as long as the level is not OK (see *Section 19.4.4*)
  - The V_{DDCPU} OK monitoring flag is available from SBF_D1 in PWR_CPU1CR, and from VCPURSTF in the RCC.
- V_{DDIO1}, V_{DDIO2}, V_{DDIO3}, $V_{DD33USB}$ and $V_{DDA18ADC}$ via peripheral voltage monitor
- V_{SW} via rst_vsw, which keeps V_{SW} domain in reset as long as the level is not OK
  - The V_{SW} monitoring has no flag. The V_{SW} registers reset value can be used.

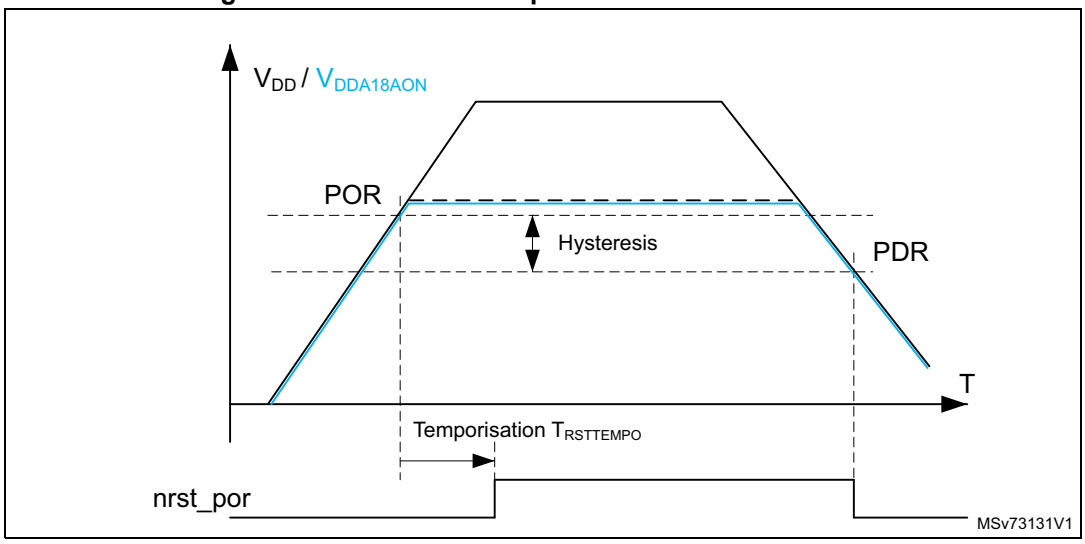
19.4.1 PWR power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that allows proper startup operation.

The system remains in reset mode when V_{DD} and $V_{DDA18AON}$ are below a specified V_{POR} threshold, without the need for an external reset circuit. Once V_{DD} and $V_{DDA18AON}$ supply levels are above the V_{POR} threshold, the system is taken out of reset. Refer to the electrical characteristics of the device datasheet for more details on the POR/PDR thresholds.

The POR/PDR generate also an application reset (NRST).

Figure 63. Power-on reset/power-down reset waveform



Note: For thresholds and hysteresis, refer to the datasheet.

19.4.2 PWR brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the V_{DD} supply reaches the specified V_{BOR} threshold, that is configured through system option bytes. By default, BOR is off. When selecting another BOR threshold, this only takes effect after the device has loaded to option bytes. The following programmable V_{BOR} thresholds can be selected:

- BOR off (V_{BOR0}): reset threshold level for above 1.67 V voltage operation
- BOR level 1 (V_{BOR1}): reset threshold level for above 2.97 V voltage operation

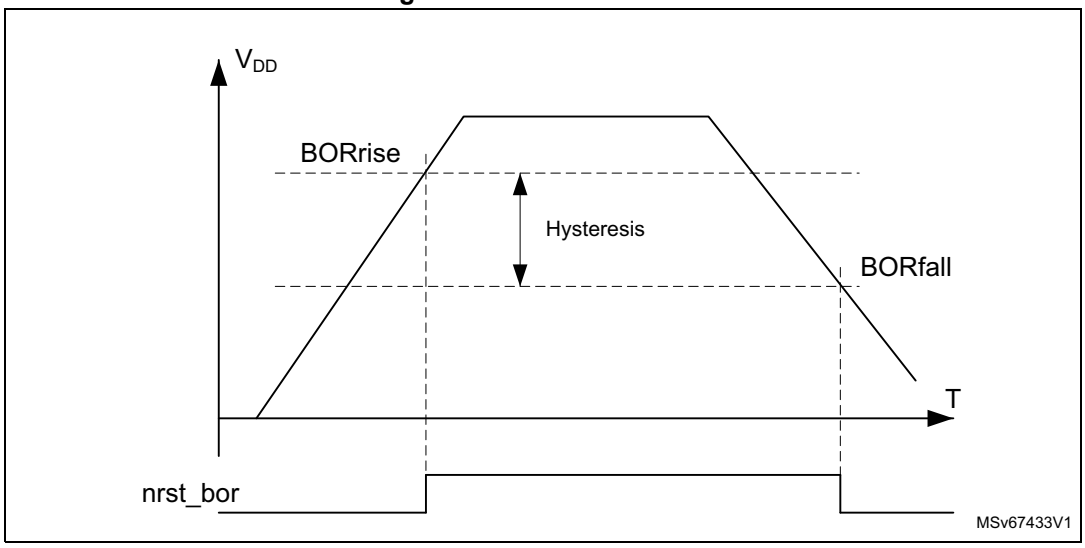
The BOR generates also an application reset (NRST).

When the BOR is enabled and the V_{DD} supply drops below the selected V_{BOR} threshold, a NRST is generated.

BOR can be disabled by programming the system option bytes. To disable the BOR function, V_{DD} must be higher than V_{BOR0} to start the system option byte programming sequence. The power down is then monitored by the PDR (see *Section 19.4.1*).

The BOR threshold hysteresis is ~100 mV for level 1, and ~40 mV for level 0 (between the rising and the falling edge of the supply voltage).

Figure 64. BOR thresholds



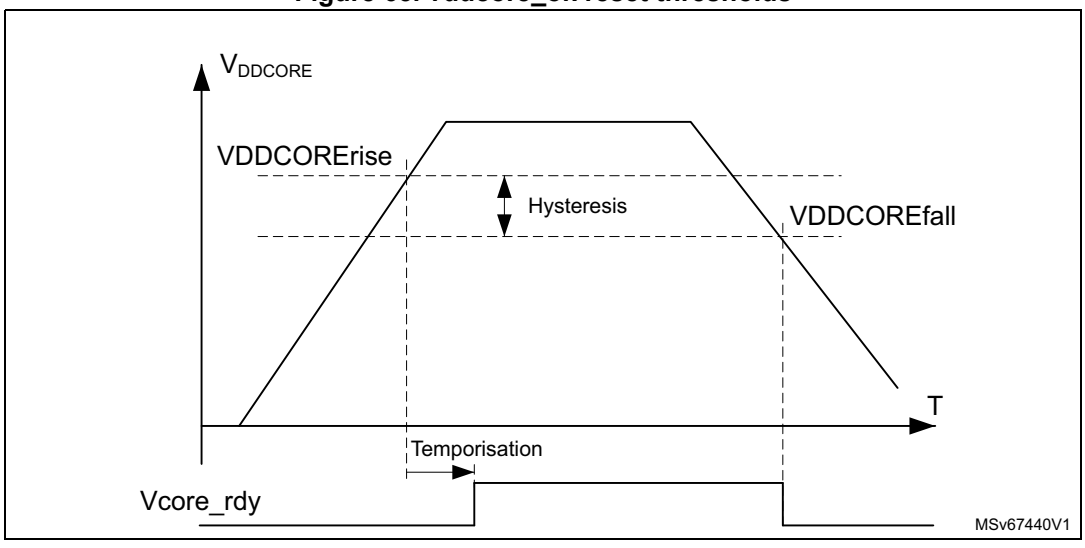
19.4.3 PWR V_{DDCORE} OK reset

The system has an integrated circuitry that allows proper startup operation of the V_{DDCORE} domain.

This domain remains in Reset mode when V_{DDCORE} is below the operation threshold vddcore_ok. Once V_{DDCORE} supply is above vddcore_ok, the V_{DDCORE} domain is taken out of reset. For more details concerning the V_{DDCORE} OK reset threshold, refer to the electrical characteristics of the device datasheet.

When LVDS_D2 is set in PWR_CPU2CR, the V_{DDCORE} supply level can be lowered in LPLV-Stop mode.

Figure 65. vddcore_ok reset thresholds



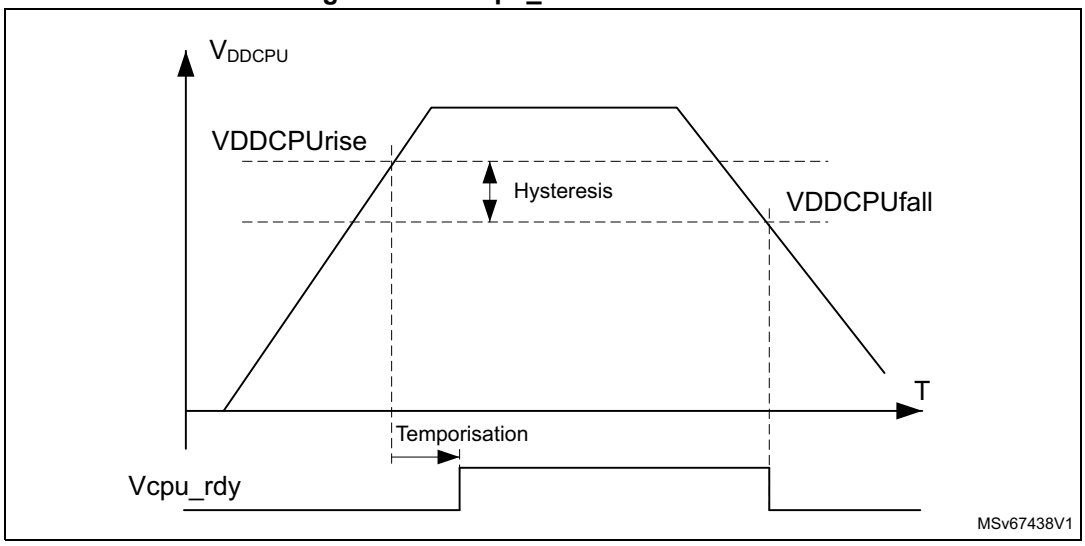
19.4.4 PWR V_{DDCPU} OK reset

The system has an integrated circuitry that allows proper startup operation of the V_{DDCPU} domain.

This domain remains in Reset mode when V_{DDCPU} is below the operation threshold vddcpu_ok. Once V_{DDCPU} is above this threshold, the V_{DDCPU} domain is taken out of reset. For more details concerning this threshold, refer to the electrical characteristics of the device datasheet.

When LVDS_D1 is set in PWR_CPU1CR, the V_{DDCPU} supply level can be lowered in LPLV-Stop mode.

Figure 66. vddcpu_ok reset thresholds



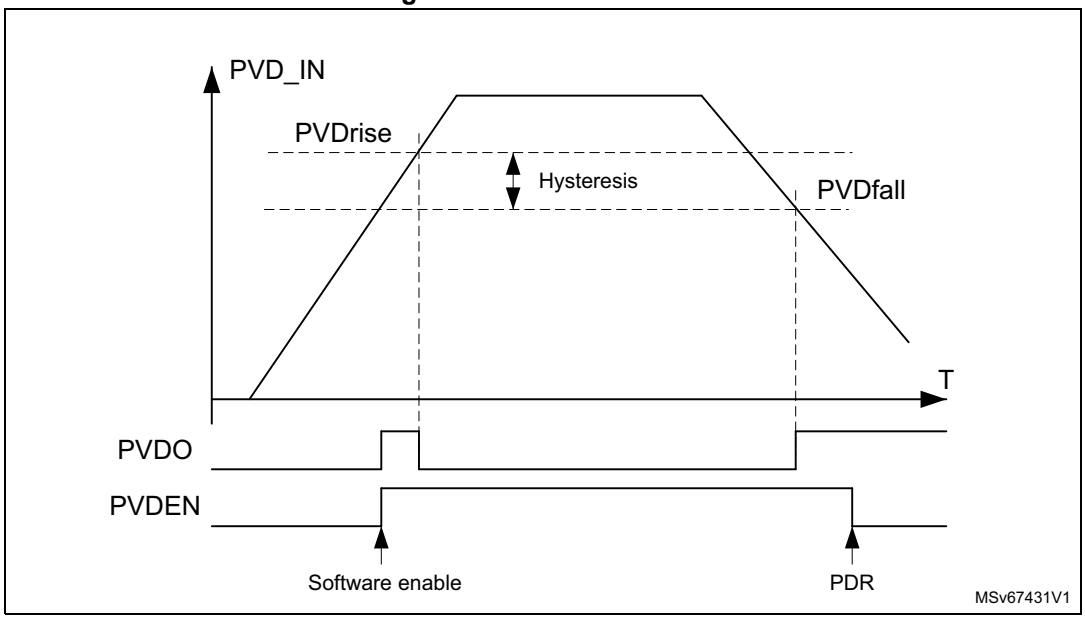
19.4.5 PWR programmable voltage detector (PVD)

The PVD can be used to monitor a voltage level on the PVD_IN pin: the voltage level on PVD_IN is compared to the internal V_{REFINT} level.

The PVD is enabled by setting PVDEN in PWR_CR3. The PVD is not available in Standby mode.

A PVDO flag in PWR_CR3 indicates if the voltage level on PVD_IN is higher or lower than the PVD threshold. This event is internally connected to the EXTI1, and can generate an interrupt if enabled through the EXTI1 registers. The PVDO output interrupt can be generated when the voltage level on PVD_IN drops below the PVD threshold, and/or when the voltage level on PVD_IN rises above the PVD threshold, depending on EXTI1 rising/falling edge configuration. As an example the service routine can perform emergency shutdown tasks.

Figure 67. PVD thresholds



Note: For thresholds and hysteresis, refer to the datasheet.

19.4.6 Peripheral voltage monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The other supplies (V_{DDIO1}, V_{DDIO2}, V_{DDIO3}, $V_{DD33USB}$ and $V_{DDA18ADC}$) can be independent from V_{DD}, and can be monitored with six peripheral voltage monitoring (PVM):

- VDDIO1VM monitors the PE[5:0] supply V_{DDIO1}. VDDIO1RDY in PWR_CR8 indicates if V_{DDIO1} is higher or lower than the V_{IO1VM} threshold.
- VDDIO2VM monitors the PE[15:6] supply V_{DDIO2}. VDDIO2RDY in PWR_CR7 indicates if V_{DDIO2} is higher or lower than the V_{IO2VM} threshold.
- VDDIO3VM monitors the PD[11:0] supply V_{DDIO3}. VDDIO3RDY in PWR_CR1 indicates if V_{DDIO3} is higher or lower than the V_{IO3VM} threshold.
- USB33VM monitors the USB supply $V_{DD33USB}$. USB33RDY in PWR_CR1 indicates if $V_{DD33USB}$ is higher or lower than the $V_{USB33VM}$ threshold.
- AVM monitors the ADC supply $V_{DDA18ADC}$. ARDY in PWR_CR1 indicates if $V_{DDA18ADC}$ is higher or lower than the V_{AVM} threshold.

Note: For thresholds and hysteresis, refer to the device datasheet.

The GPVMO flag in PWR_CR1 indicates if all enabled independent power supplies are higher or lower than the PVM threshold. This event is internally connected to the EXTI1, and can generate an interrupt if enabled through the EXTI1 registers. The GPVMO interrupt can be generated when all enabled independent power supplies rise above the PVM threshold, or when at least one enabled independent power supply drops below the PVM threshold, depending on EXTI1 line rising/falling edge configuration.

The PVM is not available in Standby mode.

The independent supplies (V_{DDIO1}, V_{DDIO2},V_{DDIO3}, $V_{DD33USB}$ and $V_{DDA18ADC}$) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies.

- If these supplies are shorted externally to V_{DD}, the application must assume they are available without enabling any peripheral voltage monitoring. The power isolation can be removed by setting the corresponding supply valid bits.
- If these supplies are independent from V_{DD}, the PVM can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB HS PHYs:

1. If $V_{DD33USB}$ is independent from V_{DD}:
   a) Enable USB33VM by setting USB33VMEN in PWR_CR1.
   b) Wait for the USB33VM wake-up time.
   c) Wait until USB33RDY is set in PWR_CR1.
   d) Optional: Disable USB33VM for consumption saving.

The following sequence must be done before using the analog to digital converters:

1. If $V_{DDA18ADC}$ is independent from V_{DD}:
   a) Enable AVM by setting AVMEN in PWR_CR1.
   b) Wait for the AVM wake-up time.
   c) Wait until ARDY is set in PWR_CR1.
2. Set ASV in PWR_CR1 to remove the $V_{DDA18ADC}$ power isolation.

The GPIOs have the capability to work in the 1.8 V or 3.3 V power supply ranges. It is needed to select the supply range of the I/Os before to use it. By default, the 3.3 V range is selected. The voltage range configuration for V_{DD} and V_{DDIO3} is retained in Standby mode. The configuration can be retained for V_{DDIO1} and V_{DDIO2} in Standby mode when VDDIO1VRSTBY and VDDIO2VRSTBY are set in PWR_CR7 and PWR_CR8. The voltage range must not be retained if the supply range can change when exiting from Standby mode (such as SD card).

The following sequence must be done before using the VDDIO1 I/Os:

1. If V_{DDIO1} is independent from V_{DD}:
   a) Enable VDDIO1VM by setting VDDIO1VMEN in PWR_CR8.
   b) Wait for the VDDIO1VM wake-up time.
   c) Wait until VDDIO1RDY is set in PWR_CR8.
   d) Optional: Disable VDDIO1VM for consumption saving.
   e) If V_{DDIO1} is in 1.8 V range, set V_{DDIO1} voltage range by setting VDDIO1VRSEL in PWR_CR8.
   f) Optional: Retain VDDIO1VRSEL configuration by setting VDDIO1VRSTBY in PWR_CR8.
2. Set VDDIO1SV in PWR_CR8 to remove the V_{DDIO1} power isolation.

The following sequence must be done before using the VDDOI2 I/Os:

1. If V_{DDIO2} is independent from V_{DD}:
   a) Enable VDDIO2VM by setting VDDIO2VMEN in PWR_CR7.
   b) Wait for the VDDIO2VM wake-up time.
   c) Wait until VDDIO2RDY is set in PWR_CR7.

d) Optional: Disable VDDIO2VM for consumption saving.

e) If V_{DDIO2} is in 1.8 V range, set V_{DDIO2} voltage range by setting VDDIO2VRSEL in PWR_CR7.

f) Optional: Retain VDDIO1VRSEL configuration by setting VDDIO2VRSTBY in PWR_CR7.

2. Set VDDIO2SV in PWR_CR7 to remove the V_{DDIO2} power isolation.

The following sequence must be done before using the VDDIO3 I/Os:

1. If V_{DDIO3} is independent from V_{DD}:
   a) Enable VDDIO3VM by setting VDDIO3VMEN in PWR_CR1.
   b) Wait for the VDDIO3VM wake-up time.
   c) Wait until VDDIO3RDY is set in PWR_CR1.
   d) Optional: Disable VDDIO3VM for consumption saving.
   e) If V_{DDIO3} is in 1.8 V range, set the V_{DDIO3} voltage range by setting VDDIO3VRSEL in PWR_CR1.
2. Set VDDIO3SV in PWR_CR1 to remove the V_{DDIO3} power isolation.

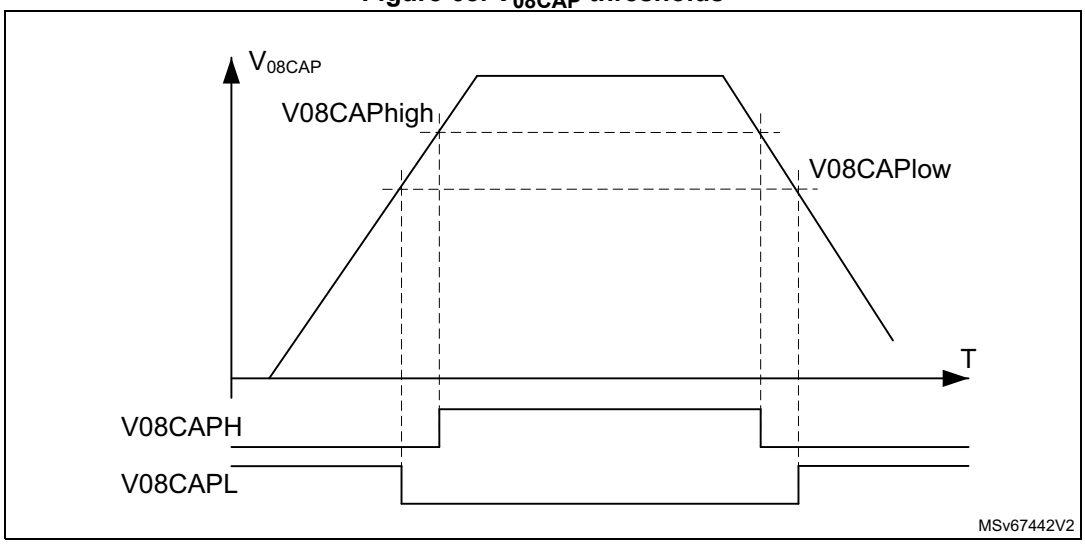
19.4.7 PWR V_{08CAP} voltage thresholds

The backup domain voltage V_{08CAP} supply can be monitored by comparing it with two threshold levels. The V08CAPH and V08CAPL flags in PWR_CR2 indicate if V_{08CAP} is higher or lower than the thresholds. These flags are available on tamper signals (see *Section 64*).

The V_{08CAP} supply monitor can be enabled/disabled with MONEN in PWR_CR2. When enabled, the V_{08CAP} voltage thresholds increase the power consumption. As an example, the levels can be used to trigger a routine to perform emergency saving tasks.

The V_{08CAP} voltage thresholds, when enabled, are also available in Standby and VBAT modes.

Figure 68. V_{08CAP} thresholds



Note: For thresholds and hysteresis, refer to the datasheet.

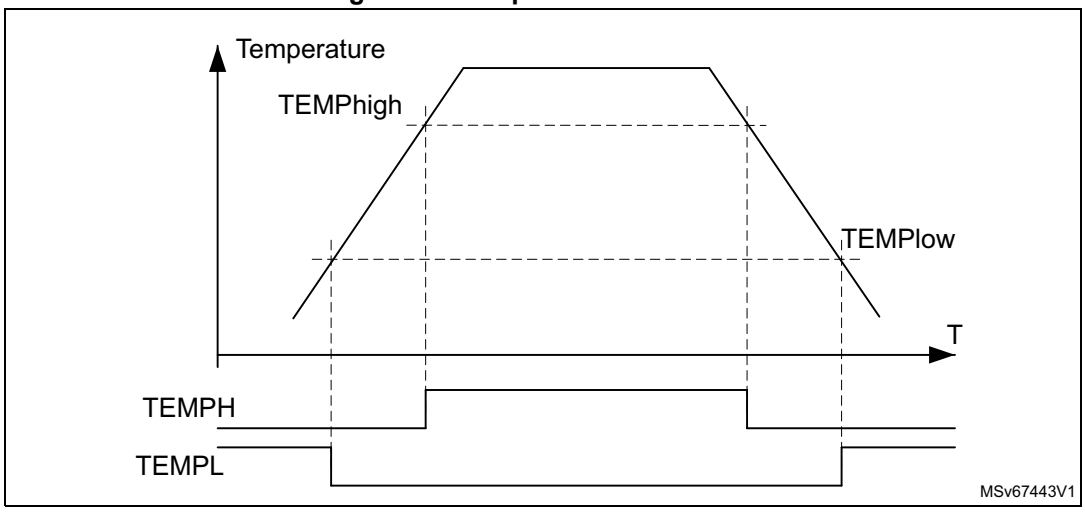
19.4.8 PWR temperature thresholds

The temperature can be monitored by comparing it with two threshold levels. The TEMPH and TEMPL flags in PWR_CR2 indicate if the device temperature is higher or lower than the threshold. These flags are available on tamper signals (see *Section 64*).

The temperature monitor can be enabled/disabled with MONEN in PWR_CR2.
When enabled, the temperature thresholds increases power consumption. As an example, the levels can be used to trigger a routine to perform temperature control tasks.

The temperature thresholds, when enabled, are also available in Standby and V_{BAT} modes.

Figure 69. Temperature thresholds



Note: For thresholds and hysteresis, refer to the datasheet.

19.5 PWR power management

The power management controls V_{DDCPU} and V_{DDCORE} supplies in accordance with the system operation modes (see *Section 19.5.1*). The system is split into three domains:

- D1 domain supplied by V_{DDCPU}, containing the Cortex-A35 (CPU1)
- system D2 domain supplied by V_{DDCORE}, containing the Cortex-M33 (CPU2), a large part of the peripherals and the system control including the EXTI1

D1 and system D2 can be in one of the following operating modes:

- DRun, Run1/2, (power On, clock On)
- DStop1/2/3, Stop1/2, LP-Stop1/2, LPLV-Stop1/2 (power on, clock off)
- DStandby, Standby (power off, clock off)

At PWR controller level, the dual-core Cortex-A35 is seen as a single CPU1.

V_{DDCORE} supply follows the system operating mode (Run, Stop, LP-Stop, LPLV-Stop and Standby).

V_{DDCPU} supply follows the domain operating mode (DRun, DStop1/2/3 and DStandby). This supply can be powered down individually when the D1 domain is in DStandby mode.

Several system operating modes are available to tune the system according to the performance needed, that is when the CPU(s) do not need to execute code and wait for an

external event. It is up to the user to select the operating mode that gives the best compromise between low-power consumption, short startup time, and available wake-up sources.

The operating modes allow the control of the clock distribution to the different system parts and the system power. The system operating mode is driven by the CPU1 subsystem and CPU2 subsystem. A CPU subsystem can include multiple domains depending on its peripheral allocation. The operating modes for the different system parts are defined below.

CPU1 and CPU2 subsystem modes

- **CRun**

  CPU and CPU subsystem peripherals according to RCC PERxEN are clocked.
- **CSleep**

  CPU clocks are stalled and CPU subsystem allocated peripheral clocks operate according to RCC PERxLPEN.

  When the CPU1 enters CSleep with WFE, the CPU1 subsystem allocated peripheral clocks operate as in CPU1 CRun mode, irrespective of RCC PERxLPEN.
- **CStop**

  CPU and CPU subsystem peripheral clocks are stalled. The peripherals in autonomous mode can be clocked (independent clock request capability).

D1 domain modes

- **DRun**

  The CPU is clocked. The domain CPU subsystem is in CRun or CSleep mode.
- **DStop1**

  LPDS_D1 in PWR_CPU1R selects Stop. The CPU clock is stalled. The domain CPU subsystem is in CStop mode and PDDS_D1 in PWR_CPU1CR selects Stop.
- **DStop2**

  LPDS_D1 in PWR_CPU1R selects LP-Stop. LVDS_D1 in PWR_CPU1R selects normal voltage.The CPU clock is stalled. The domain CPU subsystem is in CStop mode and PDDS_D1 in PWR_CPU1CR selects DStop.
- **DStop3**

  LPDS_D1 and LVDS_D1 in PWR_CPU1R select LPLV-Stop.The CPU clock is stalled. The domain CPU subsystem is in CStop mode and PDDS_D1 in PWR_CPU1CR selects DStop.
- **DStandby**

  The domain is power down.The domain CPU subsystem is in CStop mode and PDDS_D1 in PWR_CPU1CR selects DStandby.

System D2 modes

- **Run1**

  The system clock and bus matrix clocks run. CPU1 and/or CPU2 subsystems are in CRun or CSleep. The Run mode is entered after a POR reset and a wake-up from Standby.
- **Run2**

The system clock and bus matrix clocks run. The D1 domain is in power down. The CPU2 subsystem is in CRun or CSleep. The CPU1 subsystem is in CStop, and PDDS_D1 in PWR_CPU1CR selects DStandby.

- **Stop1, LP-Stop1, LPLV-Stop1**

  The system clock and D2 bus matrix clocks are stalled. CPU1 and CPU2 subsystems are in CStop, and PDDS_D1 in PWR_CPU1CR selects DStop. At least on PDDS_D2 in PWR_CPU1CR and PWR_CPU2CR select Stop mode.

- **Stop2/LP-Stop2/LPLV-Stop2**

  The system clock and D2 bus matrix clocks are stalled. The D1 domain is in power down. CPU1 and CPU2 subsystems are in CStop, and PDDS_D1 PWR_CPU1CR selects DStandby. At least on PDDS_D2 in PWR_CPU1CR and PWR_CPU2CR select Stop mode.

- **Standby**

  The system is powered down. All CPU subsystems are in CStop. PDDS_D1 in PWR_CPU1CR selects DStandby. PDDS_D2 n PWR_CPU1CR and PWR_CPU2CR select Standby.

Note: In Run modes, the power consumption can be reduced by gating the clocks to AHBx and APBx peripherals when they are unused, with PERxEN.

PWR CPU1 low-power mode summary

The CPU1 system supports the following low-power modes:

Table 88. CPU1 low-power mode summary

<table>
<tr><th>CPU1</th><th>Processor0</th><th>Processor1</th><th>L2</th><th>Entry(1)</th><th>Wake-up</th><th>clk</th><th>Processor0 clk</th><th>Processor1 clk</th><th>Periph bus clk</th></tr>
<tr><td rowspan="6">CRun</td><td>P0RUN</td><td>P1RUN</td><td rowspan="5">L2RUN</td><td>-</td><td>-</td><td rowspan="6">ON</td><td rowspan="3">ON</td><td>ON</td><td rowspan="5">ON</td></tr>
<tr><td rowspan="2">P0RUN</td><td rowspan="2">P1STOP</td><td>WFE</td><td>CPU1 processor1 interrupt, or event, or reset</td><td rowspan="2">OFF</td></tr>
<tr><td>WFI</td><td>CPU1 processor1 interrupt or reset</td></tr>
<tr><td rowspan="2">P0STOP</td><td rowspan="2">P1RUN</td><td>WFE</td><td>CPU1 processor0 interrupt, or event, or reset</td><td rowspan="3">OFF</td><td rowspan="2">ON</td></tr>
<tr><td>WFI</td><td>CPU1 processor0 interrupt, or reset</td></tr>
<tr><td rowspan="2">CSleep</td><td rowspan="3">P0STOP</td><td rowspan="3">P1STOP</td><td>WFE(2)</td><td>CPU1(3) interrupt, or event, or reset</td><td rowspan="2">OFF</td></tr>
<tr><td rowspan="2">L2STOP</td><td>STPREQ_P[1:0] = 0b11 WFI(4)</td><td>CPU1(3) interrupt or reset</td><td>OFF</td><td>ON/OFF(5)</td></tr>
<tr><td>CStop</td><td>STPREQ_P[1:0] = 0b11 WFI(4)</td><td>CPU1(3) EXTI interrupt or reset</td><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td></tr>
</table>

1. For STPREQ_P[1:0] register bits functionality refer to the RCC. WFI = wait for interrupt. WFE = wait for event.

2. At least one CPU1 processor is in WFE, and the other CPU1 processor is in WFE or WFI.
3. Wakes up the respective processor associated with the event or interrupt.
4. All CPU1 processors are in WFI.
5. CPU1 subsystem peripherals having a PERxLPEN bit operate accordingly.

PWR CPU2 low-power mode summary

Table 89. CPU2 low-power mode summary

CPU2	Entry	Wake-up	CPU2 clk	CPU2 periph bus clk
CRun	-	-	ON	ON
CSleep	Cortex-M33 SCR SLEEPDEEP = 0, WFE	Any CPU2 interrupt or event or reset	OFF	ON/OFF(1)
	Cortex-M33 SCR SLEEPDEEP = 0, WFI or return from ISR	Any CPU2 interrupt or reset		
CStop	Cortex-M33 SCR SLEEPDEEP = 1, WFE	Any CPU2 EXTI interrupt, or event, or reset		OFF
	Cortex-M33 SCR SLEEPDEEP = 1, WFI or return from ISR	Any CPU2 EXTI interrupt or reset		

1. CPU2 subsystem peripherals having a PERxLPEN bit operate accordingly.

PWR system low-power mode summary

The system low-power modes are derived from the CPU1 and CPU2 low-power modes.

Table 90. System low-power modes

System D2 domain	D1 domain	CPU1	CPU2	Wake-up	System clk	V_{DDCCPU}	V_{DDCORE}
Run1	DRun	CRun or CSleep	CRun or CSleep	See *Table 91*	ON	ON (nominal/overdrive)	ON (nominal)
			CStop				
	DStop1	CStop	CRun or CSleep				
Run2	DStandby	CStop and PDDS_D1=1				OFF	
Stop1 (1)	DStop1	CStop	CStop		OFF	ON (nominal/overdrive)	
Stop2	DStandby	CStop and PDDS_D1 = 1				OFF	
LP-Stop1 (1)	DStop2	CStop and LPDS_D1=1 (2)	CStop and LPDS_D2=1 (3)			ON (nominal/overdrive)	
LP-Stop2	DStandby	CStop and PDDS_D1=1				OFF	
LPLV-Stop1 (1)	DStop3	CStop and LPDS_D1=1 (2) and LVDS_D1=1 (4)	CStop and LPDS_D2=1 (2) and LVDS_D2=1 (5)			ON (reduced)	ON (reduced)
LPLV-Stop2	DStandby	CStop and PDDS_D1=1				OFF	
Standby	DStandby	CStop and PDDS_D1=1	CStop and PDDS_D2=1	WKUP pins, RTC alarm, RTC wake-up, RTC tamper events, or application reset (such as NRST, IWDG)	OFF	OFF	OFF

1. At least one PDDS_Dn in PWR_CPU1CR and PWR_CPU2CR selects Stop.
2. LPDS_D1 in PWR_CPU1CR selects LP-Stop.
3. LPDS_D2 in PWR_CPU2CR selects LP-Stop.
4. LVDS_D1 in PWR_CPU1CR selects LPLV-Stop.
5. LVDS_D2 in PWR_CPU2CR selects LPLV-Stop.

Table 91. Functionalities depending on system operating mode

Peripheral[1]	Run1/Run2	Stop1/2 and LP-Stop1/2		LPLV-Stop1/2		Standby		VBAT
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
CPU1	Y/-	R/-	-	R/-	-	-	-	-
CPU2	Y	R	-	R	-	-	-	-
DBG	O	O	O	R	-	-	-	-
ROM memory	Y	R	-	R	-	-	-	-
RAMCFG	O	R	-	R	-	-	-	-
SYSRAM	O	R	-	R	-	-	-	-
SRAM1	O	O[3]	-	R	-	-	-	-
BKPSRAM	O	O	-	R	-	O[2]	-	O[2]
RETSRAM	Y	O[3]	-	R	-	O[2]	-	O[2]
DDRCTRL	O	R	-	R	-	-	-	-
OCTOSPI1	O	R	-	R	-	-	-	-
OTFDEC1[13]	O	R	-	R	-	-	-	-
FMC	O	R	-	R	-	-	-	-
Backup registers	Y	R	-	R	-	R	-	R
Brownout reset (BOR)	Y	Y	Y	Y	Y	Y	Y	-
Programmable voltage detector (PVD)	O	O	O	O	O	-	-	-
Peripheral voltage monitor (PVM)	O	O	O	O	O	-	-	-
V08CAPH/V08CAPL monitoring	O	O	O	O	O	O	O	O
TEMPH/TEMPL monitoring	O	O	O	O	O	O	O	O
HPDMAx (x = 1, 2, 3)	O	O	O[4]	R	-	-	-	-
High-speed internal (HSI)	O	O[5]	-	-	-	-	-	-
High-speed external (HSE)	O	-	-	-	-	-	-	-
Low-speed internal (LSI)	O	O	-	O	-	O	-	-
Low-speed external (LSE)	O	O	-	O	-	O	-	O
Multi-speed internal (MSI)	O	O	-	-	-	-	-	-
HSE CSS clock security system	O	-	-	-	-	-	-	-
LSE CSS clock security system	O	O	O	O	O	O	O	O
HSI / MSI frequency monitoring	O	O	O	-	-	-	-	-
RTC/auto wake-up	O	O	O	O	O	O	O	O

Table 91. Functionalities depending on system operating mode (continued)

Peripheral[1]	Run1/Run2	Stop1/2 and LP-Stop1/2		LPLV-Stop1/2		Standby		VBAT
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
USBH	O	R	O	R	-	-	-	-
OTG	O	R	O	R	-	-	-	-
SDMMCx (x = 1, 2, 3)	O	R	-	R	-	-	-	-
FDCANx[13] (x = 1, 2)	O	R	-	R	-	-	-	-
ETHx (x = 1,2 [13])	O	R	O	R	-	-	-	-
USARTx (x = 1 to 7)	O	O[6]	O[6]	R	-	-	-	-
LPUART1	O	O[6]	O[6]	R	-	-	-	-
I2Cx (x = 1 to 3)	O	O[7]	O[7]	R	-	-	-	-
I3Cx (x = 1 to 3)	O	O[8]	O[8]	R	-	-	-	-
SPIx (x = 1 to 6)	O	O[9]	O[9]	R	-	-	-	-
SAIx (x = 1 to 4)	O	R	-	R	-	-	-	-
MDF1	O	O[10]	-	R	-	-	-	-
DCMI	O	R	-	R	-	-	-	-
PSSI	O	R	-	R	-	-	-	-
DCMIPP	O	R	-	R	-	-	-	-
HDP	O	R	-	R	-	-	-	-
LTDC	O	R	-	R	-	-	-	-
ADCx (x = 1, 2)	O	R	-	R	-	-	-	-
VREFBUF	O	R	-	R	-	-	-	-
DTS	O	R	O	R	-	-	-	-
STGEN[11]	O	R	-	R	-	-	-	-
TIMx (x = 1 to 8, 10 to 17)	O	R	-	R	-	-	-	-
LPTIMx (x = 1 to 5)	O	O[12]	O[12]	R	-	-	-	-
IWDGx (x = 1, 2, 3, 4)	O	O	O	O	O	O	O	-
WWDG1	O	R	-	R	-	-	-	-
RNGx (x=1, 2)	O	R	-	R	-	-	-	-
SAES	O	R	-	R	-	-	-	-
CRYPx[13] (x = 1, 2)	O	R	-	R	-	-	-	-
HASHx (x=1, 2)	O	R	-	R	-	-	-	-
CRC	O	R	-	R	-	-	-	-
MBOX1	O	R	-	R	-	-	-	-

Table 91. Functionalities depending on system operating mode (continued)

Peripheral[1]	Run1/Run2	Stop1/2 and LP-Stop1/2		LPLV-Stop1/2		Standby		VBAT
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
GPIOs	O	O	O	O	O	-	O 6 pins	-
TAMP (number of tamper pins)	12	12	O	12	O	12	O	12

1. Legend: Y = Yes (Enable), O = Optional (Disable by default. Can be enabled by software), R = data/state retained, - = not available, wake-up highlighted in gray.
2. The BKPSRAM and RETSRAM content can optionally be retained when enabled.
3. SRAM1 and RETRAM can be accessed by HPDMA in Stop1/2 and LP-Stop1/2 modes.
4. HPDMA transfers are functional and autonomous in Stop1/2, and LP-Stop1/2 modes, and generate a wake-up interrupt on transfer events.
5. Some peripherals with wake-up from Stop capability can request HSI to be enabled. In this case, HSI is woken up by the peripheral, and only feeds the peripheral which requested it. HSI is automatically put off when the peripheral does not need it anymore.
6. USART and LPUART reception and transmission are functional and autonomous in low-power modes in asynchronous, and in SPI master modes, and generate a wake-up interrupt on transfer events.
7. I^2C reception and transmission are functional and autonomous in low-power modes, and generate a wake-up interrupt on transfer events.
8. I^3C reception and transmission are functional and autonomous in low-power modes, and generate a wake-up interrupt on transfer events.
9. SPI reception and transmission are functional and autonomous in low-power modes, and generate a wake-up interrupt on transfer events.
10. MDF1 is functional and autonomous in low-power modes, and generate a wake-up interrupt on events.
11. STGEN not accessible in Stop2, LP-Stop2, LPLV-Stop2 mode since STGEN is in V_{DDCPU} power domain.
12. LPTIM is functional and autonomous in Stop mode, and generates a wake-up interrupt on events.
13. Not present on all device part numbers (see the related datasheet for more information).

19.5.1 PWR power modes

The power control handles V_{DDCORE} and V_{DDCPU} supplies for system modes. The system operating mode depends on the individual CPU subsystem modes, and on the domain modes.

D1 domain power modes

- In DRun, V_{DDCPU} is supplied.
- In DStop1, V_{DDCPU} is supplied, and the external regulator is kept in main mode.
- In DStop2, V_{DDCPU} is supplied and the external regulator is switched in low-power mode.
- In DStop3, V_{DDCPU} may be supplied at a lower level.
- In DStandby, V_{DDCPU} is switched off.

System D2 power modes

- In Run, V_{DDCORE} is supplied. V_{DDCPU} is supplied (Run1) or switched off (Run2).

- In Stop, V_{DDCORE} is supplied. V_{DDCPU} is supplied (Stop1) or switched off (Stop2).
- In LP-Stop, V_{DDCORE} is supplied and the external regulator is switched in low-power mode. V_{DDCPU} is supplied (LP-Stop1) or switched off (LP-Stop2).
- In LPLV-Stop, V_{DDCORE} can be supplied at a lower level. V_{DDCPU} can be also supplied at a lower level (LPLV-Stop1) or switched off (LPLV-Stop2).
- In Standby, V_{DDCORE} and V_{DDCPU} are switched off.

PWR_CPU_ON and PWR_ON are used to control the external regulators suppling the D1 domain (V_{DDCPU}) and the system D2 domain (V_{DDCORE}). PWR_LP is used to notify low-power Stop mode selection for both external regulators.

If the same external regulator is used to supply V_{DDCORE} and V_{DDCPU}, only PWR_ON/PWR_LP must be used.

Table 92. External regulators control

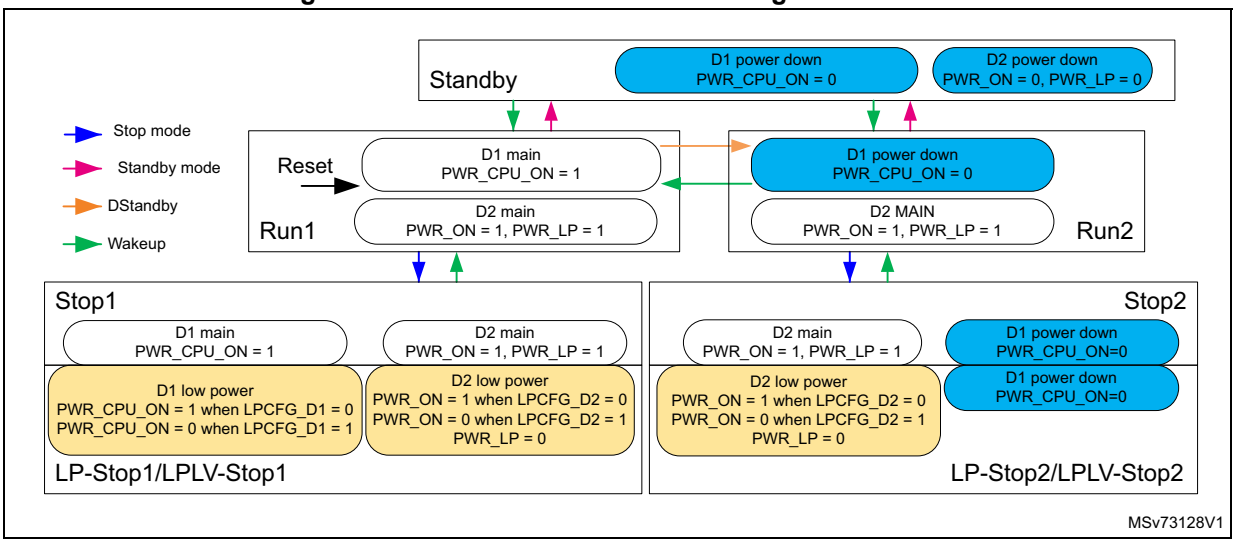
D2 system mode	D1 domain mode	PWR_CPU_ON		PWR_LP	PWR_ON	
		LPCFG_D1 = 0	LPCFG_D1 = 1		LPCFG_D2 = 0	LPCFG_D2 = 1
Run1	DRun	1	1	1	1	1
	DStop1	1	1	1	1	1
Run2	DStandby	0(1)	0(1)	1	1	1
Stop1	DStop1	1	1	1	1	1
Stop2	DStandby	0(1)	0(1)	1	1	1
LP-Stop1/LPLV-Stop1	DStop2/3	1	0	0(2)	1	0(2)
LP-Stop2/LPLV-Stop2	DStandby	0(1)	0	0(2)	1	0(2)
Standby	DStandby	0(1)	0(1)	0(3)	0(3)	0(3)

1. A guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D bits in PWR_D1CR.
2. Setting POPL_D2 bits in PWR_D2CR has no impact on LP-Stop/LPLV-Stop PWR_ON and PWR_LP pulse low time.
3. A guaranteed minimum PWR_ON and PWR_LP pulse low time can be defined by POPL_D bits in PWR_D2CR.

In LP-Stop1 and LPLV-Stop1 modes the PWR_CPU_ON is kept asserted or may be deasserted according to LPCFG_D1 in PWR_D1CR. In LP-Stop1/2 and LPLV-Stop1/2 modes, the PWR_ON is kept asserted or may be deasserted according to LPCFG_D2 in PWR_D2CR.

The definition (power off or low-power mode) of PWR_ON/PWR_CPU_ON must have been defined in the external regulator before entering LP-Stop/LPLV-Stop.

Figure 70. Power states and external regulator control

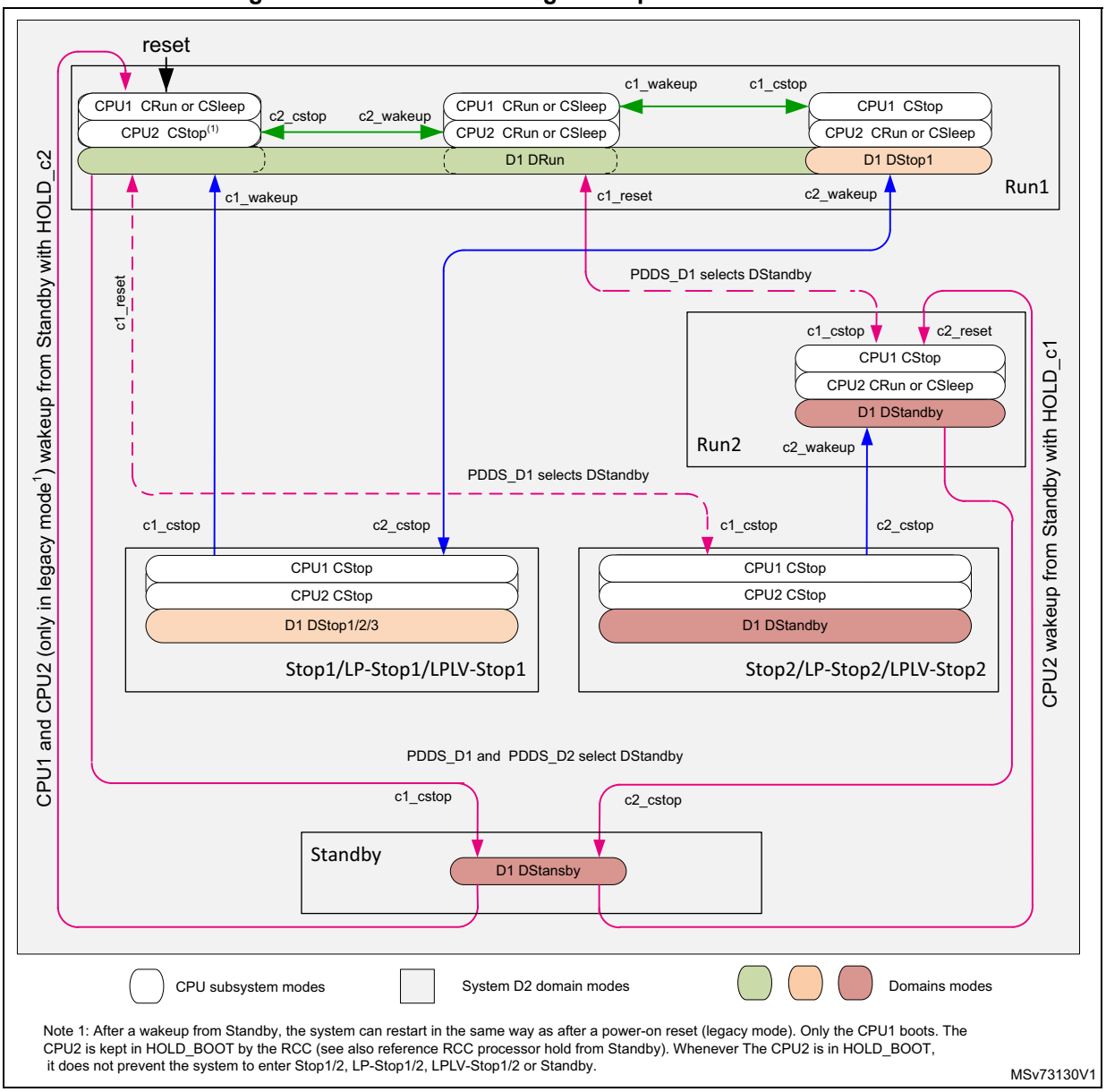


The system mode selection between Stop, LP-Stop or LPLV-Stop, and Standby is controlled by PDDS_D1 and PDDS_D2 in PWR_CPU1CR and PWR_CPU2CR. The system only enters Standby when these two bits allow so.

Table 93. PDD low-power mode control

CPU1 PDDS_D1	CPU1 PDDS_D2	CPU2 PDDS_D2	System operation mode
0	X	X	Stop1, LP-Stop1, LPLV-Stop1
1	X	0	Stop2, LP-Stop2, LPLV-Stop2
1	0	X	Stop2, LP-Stop2, LPLV-Stop2
1	1	1	Standby

Figure 71. Detailed state diagram of power control modes



Note 1: After a wakeup from Standby, the system can restart in the same way as after a power-on reset (legacy mode). Only the CPU1 boots. The CPU2 is kept in HOLD_BOOT by the RCC (see also reference RCC processor hold from Standby). Whenever The CPU2 is in HOLD_BOOT, it does not prevent the system to enter Stop1/2, LP-Stop1/2, LPLV-Stop1/2 or Standby.

MSv73130V1

The power control state transitions are initiated by (referring to *Figure 71*):

- CPU1 or CPU2 going to CStop
  - Green transitions: the system stays in Run modes (Run1 or Run2).
  - Blue line transitions: the system enters Stop1/2, LP-Stop1/2 or LPLV-Stop1/2.
  - Red transitions: the system enters Standby.
- D1 going to DStandby
  - Dotted red line transitions: The system mode entered depends also on CPU2.
    The system enters Run2 when CPU2 is in CRun or CSleep.
    The system enters Stop2, LP-Stop2 or LPLV-Stop2 when CPU2 is in CStop.

- Wake-up event
  - Green transitions: CPU1 or CPU2 wake up as from CSleep.
  - Blue transitions: CPU1 or CPU2 wake up from CStop. STOPF is set in PWR_CPU1CR and PWR_CPU2CR.
  - Dotted red line transitions: D1 wakes up from DStandby. SBF_D1 is set in PWR_CPU1CR.

    When the system remains in Run modes, no additional flags are set.

    When the system wakes up from Stop2, LP-Stop2 or LPLV-Stop2, STOPF is also set in PWR_CPU1CR.
  - Red transitions: CPU1 and CPU2 wake up from Standby. SBF_D1/2 are set in PWR_CPU1CR and PWR_CPU2CR.

The flags provided to determine from which mode the system exits are shown in the table below. CPU1 and CPU2 have their own set of flags which can be read from PWR_CPU1CR and PWR_CPU2CR. CPU1 and CPU2 have their own reset bit CSSF to clear their wake-up flags. CPU1 and CPU2 CSSF bits are located in the same register as CPU1 and CPU2 flags.

Table 94. Low-power mode exit flags

System D2 mode	D1 domain	SBF_D1	SBF_D2	SBF	STOPF	VBF	Comment
Run1	DRun/DStop1	0	0	0	0	0	Retained system contents
Run2	DStandby	1	0	0	0	0	Lost D1 contents. Retained system contents.
Stop1 LP-Stop1 LPLV-Stop1	DStop1/2/3	0	0	0	1	0	Retained system contents. System clock stopped.
Stop2 LP-Stop2 LPLV-Stop2	DStandby	1	0	0	1	0	Lost D1 contents. Retained system contents. System clock stopped.
Standby	DStandby	1	1	1	0	0	Lost system contents
V_{BAT}	NA	0	0	0	0	1	

19.5.2 Low-power batch acquisition mode (LPBAM)

The devices support Low-power batch acquisition mode (LPBAM), which allows peripherals to be functional and autonomous in low-power modes (without any software running).

Stop1/2, LP-Stop1/2 modes

In Stop1/2 and LP-Stop1/2 modes, the autonomous peripherals are: LPTIMx (x = 1 to 5), USARTx (x = 1 to 7), LPUART1, SPIx (x = 1 to 6), I2Cx (x = 1 to 3), I3Cx (x = 1 to 3), MDF1 and HPDMAx (x = 1 to 3). In these modes, SRAM1 and RETRAM can be accessed by the HPDMAx. GPIOA-I,Z can be also accessed by the HPDMAx.

LPBAM features

These autonomous peripherals support the following features:

- Functionality in Stop mode thanks to their own independent clock (named kernel clock) request capability: the peripheral kernel clock is automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it.
- DMA transfers supported in Stop mode thanks to system clock request capability: the system clock (HSI) automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it. When the system clock is requested by an autonomous peripheral, the system clock is woken up and distributed to all autonomous peripherals, and to all GPIOs enabled in the RCC. This allows the DMA to access the enabled SRAM, and any enabled peripheral register (for instance any autonomous peripherals or GPIO registers).
- Automatic start of the peripheral thanks to hardware synchronous or asynchronous triggers (such as I/Os edge detection and low-power timer event).
- Wake-up from Stop mode with peripheral interrupt: the HPDMAs are fully functional and the linked-list is updated in Stop mode, allowing the different DMA transfers to be linked without any CPU wake-up. This can be used to chain different peripheral transfers, or to write peripheral registers in order to change their configuration while remaining in Stop mode.

  The DMA transfers from memory-to-memory can be started by hardware synchronous or asynchronous triggers. The DMA transfers between peripherals and memories can also be gated by these triggers.

Here below some examples of use-cases that can be done while remaining in Stop mode:

- I^2C target reception or transmission, SPI reception, UART/LPUART reception
  - wake-up at the end of peripheral transfer or on DMA buffer event
- I^2C controller transfer, SPI transmission, UART/LPUART transmission, triggered by a low-power trigger (or any other trigger)
  - example: sensor periodic read
  - wake-up at the end of peripheral transfer or on DMA buffer event
- Data transfer from/to GPIO to/from SRAM:
  - to control external components
  - to implement data transmission and reception protocols

19.5.3 PWR system wake-up

From an application reset (such as NRST PAD, IWDG, POR, BOR), the system wakes up the CPU1. The CPU2 is kept in HOLD_BOOT. From this point, the CPU1 boot ROM manages the startup (see *Section 20*).

From Stop1/2, LP-Stop1/2, LPLV-Stop1/2, and Standby, the system can be configured to wake up CPU1 or CPU2 independently.

19.6 PWR low-power modes

Several low-power modes are available to save power when CPU1 and/or CPU2 do not need to execute code (when waiting for an external event). It is up to the user to select the

mode that gives the best compromise between low-power consumption, short startup time and available wake-up sources:

- slowing down system clocks (see *Section 20*)
- controlling individual peripheral clocks (see *Section 20*)
- low-power modes:
  - CSleep (CPU clock stopped)
  - CStop (CPU subsystem clock stopped)
  - D1 DStop1 (CPU subsystem clock stopped, normal mode signaled to external regulator)
  - D1 DStandby (domain power down and wake-up via reset)
  - Stop1, LP-Stop1, and LPLV-Stop1 (system clock stalled, normal or low-power mode signaled to external regulator supplying V_{DDCPU} and V_{DDCORE})
  - Stop2, LP-Stop2, and LPLV-Stop2 (system clock stalled, powered-down mode signaled to external regulator supplying V_{DDCPU}. Normal or low-power mode signaled to external regulator supplying V_{DDCORE}).
  - Standby (system domain in power down)

19.6.1 PWR slowing down CPU and/or bus matrix clocks

In Run mode, the speed of CPU1, CPU2 and/or bus matrix clocks can be reduced (see *Section 20* for more details).

19.6.2 PWR controlling peripheral clocks

In Run mode, the bus clock for individual peripherals can be stopped by RCC PERxEN register bits at any time to reduce power consumption.

To reduce power consumption in CSleep mode, the individual peripheral clocks can be disabled by the RCC PERxLPEN register bit. For the peripherals still receiving a clock in CSleep mode, their clock can be slowed down before entering CSleep mode.

19.6.3 PWR entering low-power modes

The CPU1 subsystem low-power modes CSleep and CStop are entered by the CPU1 when executing WFI or WFE instructions.

The CPU2 subsystem low-power modes CSleep and CStop are entered by the CPU2 when executing WFI or WFE instructions, or when SLEEPONEXIT in Cortex-M33 system control register is set on Return from ISR.

The D1 domain can enter in DStandby when all EXTI wake-up sources assigned to CPU1 are cleared, and the CPU1 is in CStop.

The system enters to Stop1/2, LP-Stop1/2, LPLV-Stop1/2, or Standby when all EXTI wake-up sources are cleared, and both CPU1 and CPU2 are in CStop.

The system enters in Standby when all EXTI wake-up sources are cleared, and all CPUs are in CStop.

19.6.4 PWR exiting from low-power modes

From CSleep mode, CPU1 and CPU2, subsystems exit from low-power mode through any interrupt or event, depending on how the low-power mode was entered.

If the WFI instruction or Return from ISR was used to enter low-power mode, any peripheral interrupt acknowledged by the GIC/NVIC can wake up the system.

If the WFE instruction is used to enter low-power mode, CPU1 and CPU2 exit from low-power mode as soon as an event occurs. The wake-up event can be generated by one of the following ways:

- GIC/NVIC IRQ interrupt
  - For CPU1, enable an interrupt in the peripheral control register and in the GIC.
  - Only GIC interrupts with sufficient priority wakes up and interrupt the CPU1.
  - For CPU2, when SEVONPEND = 0 in Cortex-M33 system control register, enable an interrupt in the peripheral control register and in the NVIC. When CPU2 resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
  - Only NVIC interrupts with sufficient priority wake up and interrupt CPU2.
  - For CPU2, when SEVONPEND = 1 in Cortex-M33 system control register, enable an interrupt in the peripheral control register and optionally in the NVIC. When CPU2 resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

    All NVIC interrupts wake up CPU2 , even the disabled ones.

    Only enabled NVIC interrupts with sufficient priority wake up and interrupt the CPU2.
- Event

  For CPU1 and CPU2 configuring an EXTI line in event mode, when CPU1 or CPU2, resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the GIC/NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It may be necessary to clear the interrupt flag in the peripheral.

From CStop, when the system is in Stop1, LP-Stop1, and LPLV-Stop1 modes, CPU1 and/or CPU2 subsystems exit from low-power mode only through an enabled EXTI interrupt or event, depending on how the low-power mode was entered.

From CStop, when the system is in Run2, Stop2, LP-Stop2, and LPLV-Stop2, the CPU1 subsystem exits from low-power mode only through an enabled EXTI interrupt.The program execution restarts with a CPU1 reset. The CPU2 subsystem exits from low-power mode through an enabled EXTI interrupt or event, depending on how the low-power mode was entered.

From Standby, CPU1 and/or CPU2 subsystems exit from low-power mode only through an application reset (such as NRST, IWDG), enabled WKUPx pins and a RTC event.

In legacy mode, the program execution restarts in the same way as after a power-on reset (such as reset vector fetched). An event, if enabled, can wake up the CPU2 and kept the D1 domain in DStandby. The CPU2 firmware is in charge to restore the secure context of the system before executing application restart.

19.6.5 PWR CSleep mode

The CSleep mode applies only to the CPU subsystems. In CSleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the PERxLPEN bits.

PWR entering to CSleep mode

The CSleep mode is entered according to *Section 19.6.3*:

- For CPU1, one of the STPREQ_P[1:0] register bits in RCC is cleared (see *Table 95*).
- For CPU2, SLEEPDEEP bit in Cortex-M33 system control register is cleared (see *Table 96*).

PWR exiting from CSleep mode

The CSleep mode is exited according to *Section 19.6.4* and the tables below.

Table 95. CPU1 CSleep mode

CPU1 CSleep	Comments
Mode entry	WFI while: – STPREQ_P[1:0] = 0b11 (see *Section 20*). – CPU1 GIC interrupts and events are cleared.
	WFE while CPU1 GIC interrupts and events cleared.
Mode exit	If WFI was used for entry, any interrupt enabled in GIC (see *Section 28.4).* If WFE was used for entry, any interrupt event when disabled in GIC (see *Section 28.4),* or any event (see *Section 19.6.4)*
Wake-up latency	None

Table 96. CPU2 CSleep mode

CPU2 CSleep	Comments
Mode entry	WFI or WFE while: – SLEEPDEEP = 0 (see the Cortex-M33 system control register). – CPU2 NVIC interrupts and events are cleared.
	On return from ISR while: – SLEEPDEEP = 0 and SLEEPONEXIT = 1 (see the Cortex-M33 system control register). – CPU2 NVIC interrupts and events are cleared.
Mode exit	If WFI or return from ISR was used for entry, any interrupt enabled in NVIC (see *Section 27.3*) If WFE was used for entry and SEVONPEND = 0, any event (see *Section 19.6.4)* If WFE was used for entry and SEVONPEND = 1, any interrupt event disabled in NVIC (see *Section 27.3*), or any event (see *Section 19.6.4).*
Wake-up latency	None

19.6.6 PWR CStop mode

The CStop mode applies only to the CPU subsystems. In CStop mode, the CPU clock and CPU subsystem peripheral clocks are stopped.

In CStop mode, CPU subsystem peripherals having a kernel clock request are still able to request their kernel clock, when selecting a clock in the RCC, which is able to run in CStop mode.

PWR entering to CStop mode

The CStop mode is entered according to *Section 19.6.3:*

- For CPU1, when STPREQ_P[1:0] in RCC are set (see *Table 97*).
- For CPU2, when SLEEPDEEP in Cortex-M33 system control register is set (see *Table 98*)

PWR exiting from CStop mode

The CStop mode is exited according to *Section 19.6.4* and the tables below.

Table 97. CPU1 CStop mode

CPU1 CStop	Comments
Mode entry	WFI while: – STPREQ_P[1:0] = 0b11 (see *Section 20*) – CPU1 GIC interrupts and events are cleared. – All CPU1 EXTI1-2 wake-up sources are cleared.
Mode exit	EXTI1-2 interrupt is enabled in GIC (see *Section 28.4)* for peripherals which are not stopped.
Wake-up latency	EXTI and RCC wake-up synchronization (see *Section 20*)

Table 98. CPU2 CStop mode

CPU2 CStop	Comments
Mode entry	WFI or WFE while: – SLEEPDEEP = 1 (see Cortex-M33 system control register). – CPU2 NVIC interrupts and events are cleared. – All CPU2 EXTI1-2 wake-up sources are cleared.
	On return from ISR while: – SLEEPDEEP = 1 and SLEEPONEXIT = 1 (see Cortex-M33 system control register). – CPU2 NVIC interrupts and events are cleared. – All CPU2 EXTI1-2 wake-up sources are cleared.
Mode exit	If WFI or return from ISR was used for entry, EXTI1-2 interrupt enabled in NVIC (see *Section 27.3*) for peripherals which are not stopped If WFE was used for entry and SEVONPEND = 0, EXTI1-2 event (see *Section 19.6.4*) for peripherals which are not stopped. If WFE was used for entry and SEVONPEND = 1, EXTI1-2 interrupt event when disabled in NVIC (see *Section 27.3*), or EXTI event (see *Section 19.6.4*) for peripherals which are not stopped
Wake-up latency	EXTI and RCC wake-up synchronization. (see *Section 20*)

19.6.7 PWR D1 DStop1 mode

In DStop1 mode, the domain CPU clock is stalled.

PWR entering to DStop1 mode

The DStop1 mode is entered according to *Section 19.6.3* when PDDS_D1 in PWR_CPU1CR selects Stop for the D1 domain.

See *Table 99* for details on how D1 domain enters DStop1.

PWR exiting DStop1 mode

The DStop1 mode is exited according to *Section 19.6.4.* See the table below for more details on how D1 domain exit DStop1. When exiting DStop1 mode, the CPU subsystem clock is the same as when entering DStop1.

Table 99. D1 DStop1 mode

D1 DStop1	Comments
Mode entry	– When CPU1 subsystem is in CStop mode. – PDDS_D1 = 0 in PWR_CPU1CR – LPDS_D1 = 0 in PWR_CPU1CR.
Mode exit	On an EXTI1-2 wake-up
Wake-up latency	Clocks restored, and EXTI and RCC wake-up synchronization (see *Section 20*)

19.6.8 PWR D1 DStandby mode

Like the DStop1 mode, the DStandby mode is based on CPU subsystem CStop mode, but the domain supply is powered off.

In DStandby, the domain is powered down and the domain RAM and register contents are lost.

Entering D1 DStandby mode

The D1 DStandby mode is entered according to *Section 19.6.3* when PDDS_D1 in PWR_CPU1CR selects Standby.

Refer to *Table 100* for details on how D1 domain enters DStandby mode.

Exiting D1 DStandby mode

The DStandby mode is exited according to *Section 19.6.4*. See the table below for more details on how D1 domain exit DStandby.

When exiting D1 DStandby mode, the domain CPU and peripherals are reset. The state of the CPU subsystem clocks depends on the system mode:

- When the system did not enter Standby mode, the CPU subsystem clocks are the same as when entering DStandby.
- When the system entered Standby mode, the CPU subsystem clocks are reset.

Table 100. D1 DStandby mode

D1 DStandby	Comments
Mode entry	– When CPU1 subsystem is in CStop and there is no active EXTI1/2 wake-up source assigned to CPU1 – PDDS_D1 = 1 in PWR_CPU1CR.
Mode exit	– On an EXTI1-2 wake-up when the system did not enter in Standby – Otherwise WKUP pins, RTC alarm (Alarm A and Alarm B), RTC wake-up, tamper event, time stamp event, application reset (such as NRST, IWDG)
Wake-up latency	– EXTI and RCC wake-up synchronization (see *Section 20*) – Domain power up and CPU1 reset – CPU1 reboot from boot ROM

19.6.9 PWR Stop1 mode

The system enters Stop1 mode only when both CPU1 and CPU2 subsystems are in CStop, and EXTI1/2 wake-up sources are inactive. In Stop1 mode, the system clock including any PLL, and the system bus matrix clocks are stopped. The system oscillators operate according to the xxxKERON register bit.

In Stop1 mode, peripherals using the LSI or LSE clock, and peripherals having a kernel clock request, selecting a clock source in the RCC which is able to run in Stop1 mode, are still able to operate.

In system Stop1 mode, the following features can be selected to be kept active by programming individual control bits:

- Independent watchdog: the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except by an application reset (see *Section 61*).
- Real-time clock: the RTC is configured by RCC_RTCCFGR.
- Internal RC oscillator: the LSI RC is configured by LSION in RCC_BDCR.
- External 32.768 kHz oscillator: the LSE OSC is configured by LSEON in RCC_BDCR.
- Peripherals can run on the LSI or LSE clock.
- Peripherals have a kernel clock request.
- Internal RC oscillators: The HSI, MSI, and HSE oscillators are configured by the corresponding xxxKERON bit in the *Section 20: Reset and clock control (RCC)*
- The ADCs can also consume power during Stop mode, unless they are disabled before entering this mode. To disable them, ADON bits in ADC_CR2 register must both be written to 0.

PWR entering to Stop1 mode

The Stop1 mode is entered according to *Section 19.6.3* when PDDS_D1 in PWR_CPU1CR requests Stop, and at least one of PDDS_D2 bits in PWR_CPU1CR and PWR_CPU2CR requests Stop.

Refer to *Table 101* for details on how to enter Stop1 mode.

PWR exiting from Stop1 mode

The Stop mode is exited according to *Section 19.6.4*. See *Table 101* for more details on how to exit Stop1 mode.

When exiting Stop mode, the RCC restores the CPU subsystem and matrix clocks. STOPF flags in PWR_CPU1CR and PWR_CPU1CR indicate that the system has exit Stop.

Table 101. Stop1 mode

Stop1	Comments
Mode entry	– When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source – PDDS_D1 = 0 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D1 = 0 in PWR_CPU1CR or LPDS_D2 = 0 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clocks restored, EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*)

PWR I/O states in Stop1 mode

In Stop1 mode, all I/O pins keep their configuration.

19.6.10 PWR Stop2 mode

The Stop2 mode is used to save D1 domain power consumption. This mode is equivalent to Stop1 mode in term of available features, but with the D1 in DStandby mode.

The system enters Stop2 mode only when both CPU1 and CPU2 subsystems are in CStop and the EXTI1/2 wake-up sources are inactive.

PWR entering Stop2 mode

The Stop2 mode is entered according to *Section 19.6.3* when PDDS_D1 in PWR_CPU1CR requests Standby, and at least one of PDDS_D2 bits in PWR_CPU1CR or PWR_CPU2CR requests Stop.

Refer to *Table 102* for details on how to enter Stop2 mode.

PWR exiting from Stop2 mode

The Stop2 mode is exited according to *Section 19.6.4*. See the table below for more details on how to exit Stop2 mode.

When exiting Stop2 mode, the RCC restores the CPU subsystem and matrix clocks. SBF_D1 in PWR_CPU1CR, and STOPF in PWR_CPU1CR and PWR_CPU2CR indicate that the system has exit Stop2 mode.

Table 102. Stop2 mode

Stop2	Comments
Mode entry	– When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1/2 wake-up source – PDDS_D1 = 1 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D2 = 0 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clocks restored, EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*) D1 Domain power up and CPU1 reset, CPU1 reboot from boot ROM

PWR I/O states in Stop2 mode

In Stop2 mode, all I/O pins keep their configuration.

19.6.11 PWR LP-Stop1 and LPLV-Stop1 modes

The LP-Stop1 mode is handled in the same way as Stop1 mode, when LP-Stop mode is selected by LPDS_D1 in PWR_CPU1CR and LPDS_D2 in PWR_CPU2CR. See the table below for more details on how to enter and exit LP-Stop1 mode.

Table 103. LP-Stop1 mode

LP-Stop1	Comments
Mode entry	– When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1/2 wake-up source – PDDS_D1 = 0 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D1 = 1 in PWR_CPU1CR and LPDS_D2 = 1 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clock restored, LP-Stop timeout (external regulator to switch to main mode), EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*)

LVDS_D1 in PWR_CPU1CR and LVDS_D2 in PWR_CPU2CR are used to select whether V_{DDCPU} and V_{DDCORE} supplies during LP-Stop1 mode are kept at Run mode level, or lowered (LP-Stop1 with low-voltage mode).

In LP-Stop1 with low-voltage mode peripherals in D2 domain that can wake up from Stop1 (such as TIM, UART, I2C, SPI), must all be disabled. The DDR retention must be enabled by

DDRRETEN in PWR_CR11. See the table below for more details on how to enter and exit LP-Stop1 with low-voltage mode.

Table 104. LPLV-Stop1 mode

LPLV-Stop1	Comments
Mode entry	– When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1/2 wake-up source – PDDS_D1 = 0 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D1 = 1 in PWR_CPU1CR and LPDS_D2 = 1 in PWR_CPU2CR – LVDS_D1 = 1 in PWR_CPU1CR and LVDS_D2 = 1 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clock restored, LPLV-Stop timeout (external regulator to restore Run mode operating supply level), EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*)

19.6.12 PWR LP-Stop2 and LPLV-Stop2 modes

The Lp-Stop2 mode is used to save D1 domain power consumption. This mode is equivalent to LP-Stop1 mode but with the D1 in DStandby mode.

The LP-Stop2 mode is handled in the same way as Stop2 mode, when LP-Stop mode is selected by LPDS_D2 register bit in PWR_CPU2CR.

Refer to the table below for more details on how to enter and exit LP-Stop2 mode.

Table 105. LP-Stop2 mode

LP-Stop2	Comments
Mode entry	– When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1/2 wake-up source – PDDS_D1 = 1 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D2 = 1 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clock restored, LP-Stop timeout (external regulator to switch to main mode), EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*) D1 domain power up and CPU1 reset, CPU1 reboot from boot ROM

LVDS_D2 in PWR_CPU2CR selects whether V_{DDCORE} supply during LP-Stop2 mode is kept at Run mode level, or lowered (LP-Stop2 with low-voltage mode).

In LP-Stop2 with low-voltage mode the peripherals in D2 domain that can wake up from Stop2 (such as TIM, UART, I2C, SPI), must all be disabled. The DDR retention must be enabled by DDRRETEN in PWR_CR11. See the table below for more details on how to enter and exit LP-Stop2 with low-voltage mode.

Table 106. LPLV-Stop2 mode

LPLV-Stop2	Comments
Mode entry	– When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1/2 wake-up source – PDDS_D1 = 1 in PWR_CPU1CR – PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR – LPDS_D2 = 1 in PWR_CPU2CR – LVDS_D2 = 1 in PWR_CPU2CR
Mode exit	On an EXTI1/2 wake-up
Wake-up latency	Clock restored, LPLV-Stop timeout (external regulator to restore Run mode operating supply level), EXTI and RCC wake-up synchronization and PWRLP_TEMPO (see *Section 20*) D1 domain power up and CPU1 reset, CPU1 reboot from boot ROM

19.6.13 PWR Standby mode

The Standby mode is used to achieve the lowest power consumption. It is based on CPU1 and CPU2 subsystem CStop mode, as for Stop modes, but with V_{DDCORE} and V_{DDCPU} supply regulators powered off.

When the system enters Standby, PWR_ON, PWR_CPU_ON and PWR_LP pins are set low. The external voltage regulators that provide V_{DDCORE} and V_{DDCPU} supplies, are disabled. Complete V_{DDCORE} and V_{DDCPU} domains are consequently powered off. PLLs, HSI and HSE oscillators are also switched off. SRAM and register contents are lost except for V_{DD}, V_{SW}, backup domain, and retention domain (RTC registers, RTC backup register, LSE, backup RAM and retention RAM) and Standby circuitry.

In system Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog: the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by an application reset (see *Section 61*).
- Real-time clock: the RTC is configured by RCC_RTCCFGR.
- Internal RC oscillator: the LSI RC is configured by LSION in RCC_BDCR.
- External 32.768 kHz oscillator: the LSE OSC is configured by LSEON in RCC_BDCR

PWR entering Standby mode

The Standby mode is entered according to *Section 19.6.3* when all PDDS_Dn bits in PWR_CPU1CR and PWR_CPU2CR request Standby mode.

Refer to *Table 108* for more details on how to enter Standby mode.

PWR exiting from Standby mode

The Standby mode is exited according to *Section 19.6.4*. See the table below for more details on how to exit Standby mode.

The system exits Standby when an application reset (such as NRST, IWDG), a WKUP pin event is detected. PWR registers are reset after wake-up from Standby, except for power control and status registers (reset condition defined for each PWR register).

Wake-up event source (WKUP pins and RTC/TAMP when the resource isolation framework is used) can be assigned independently to CPU1 and/or CPU2. The external regulators that provide V_{DDCPU} and V_{DDCORE} are enabled according to the CPU to wake up but also according to the first CPU allowed to boot. Refer to RCC Processor state after Standby exit for more details about the first CPU allowed to boot.

The selection of the CPU to wake-up is notified to the RCC through PWR CPU1_HW_BEN and CPU2_HW_BEN signals. Once the wake-up source is cleared, CPU1_HW_BEN and CPU2_HW_BEN are cleared to 0.

Table 107. CPU1_HW_BEN and CPU2_HW_BEN management

Standby exit event	First CPU allowed to boot	CPU1_HW_BEN	CPU2_HW_BEN	PWR_CPU_ON	PWR_ON
CPU1 wake-up event	CPU1	1	0	1	1
	CPU2	1	0	1	1
CPU2 wake-up event	CPU1	0	1	1	1
	CPU2	0	1	0	1
CPU1and CPU2 wake-up event	CPU1	1	1	1	1
	CPU2	1	1	1	1

SBF_D1, SBF_D2 and SBF flags in PWR_CPU1CR and PWR_CPU2CR indicate that the system has exit Standby.

Table 108. Standby mode

Standby	Comments
Mode entry	– All CPU subsystems are in CStop and there is no active EXTI1/2 wake-up source – All WKUPF bits in PWR_WKUPCRx are cleared. – All RTC wake-up events are cleared. – PDDS_D1=1 in PWR_CPU1CR – PDDS_D2=1 in both PWR_CPU1CR and PWR_CPU2CR
Mode exit	WKUP pins, RTC alarm (Alarm A and Alarm B), RTC wake-up, tamper event, timestamp event, application reset (such as NRST, IWDG(1))
Wake-up latency	System reset phase (see *Section 20*) or CPU1 boot from boot ROM or CPU2 boot from RETRAM

1. The IWDG1/2/3/4 when configured to generate an application reset, and when configured to generate a system debug wake-up event, reset the system in both cases.

PWR I/O states in Standby mode

In Standby mode, all I/O pins are high impedance no pulls except for:

- NRST, PWR_ON, PWR_CPU_ON, and PWR_LP pins (still available)
- RTC_AF1 pin if configured for tamper, timestamp, RTC Alarm out, or RTC clock calibration out
- WKUP pins, if enabled (WKUP pin pulls can be configured by WKUPPUPD in PWR_WKUPCRx)

19.7 PWR VBAT operation

The VBAT pin is used to power the V_{BAT} domain from an external battery.

The VBAT pin supplies the RTC with LSE, anti-tamper detection (TAMP), backup registers and 8-Kbyte backup SRAM. Sixteen anti-tamper detection pins are available in V_{BAT} mode.

The device can be set in V_{BAT} mode by powering down the V_{DD} and $V_{DDA18AON}$ supplies. The V_{DDCPU} and V_{DDCORE} supplies must be powered off before V_{DD} and $V_{DDA18AON}$ (PWR_ON and PWR_CPU_ON low level).

VBF flag in PWR_CPU1CR and PWR_CPU2CR indicate that the system has exit V_{BAT} mode.

PWR I/O states in V_{BAT} mode

In V_{BAT} mode, all I/O pins are powered down except for RTC and tamper pins.

19.7.1 PWR debug Stop mode

When the system enters Stop, its operation depends on the DBG_STOP configuration in DBGMCU_CR.

When the Stop mode debug is enabled, when entering Stop mode, the PWR does not signal low-power mode on PWR_ON, PWR_CPU_ON and PWR_LP signals. V_{DDCORE} and V_{DDCPU} domains are kept supplied by the regulator in main mode.

When exiting Stop mode, the system CPU and logic are enabled as from normal Stop mode. STOPF is set in PWR_CPU1CR.

When the Stop mode debug is enabled, when entering Run2 or Stop2 mode, the PWR does not signal power-down mode on the PWR_CPU_ON signal. V_{DDCPU} domain is kept supplied by the regulator in main mode.

When exiting Run2 or Stop2 mode, a CPU1 reset is generated, which resets the V_{DDCPU} logic excluding the debug logic. SBF_D1 is set in PWR_CPU1CR.

19.7.2 PWR debug Standby mode

When the system enters low-power mode Standby, its operation depends on the DBG_STANDBY configuration in DBGMCU_CR.

When the Standby mode debug is enabled, when entering Standby mode, the PWR keeps pwr_on and pwr_cpu_on active, allowing V_{DDCORE} and V_{DDCPU} domains to remain supplied. In Standby mode, V_{DDCORE} and V_{DDCPU} domains are not reset, allowing the CPU and bus clocks to be active.

When exiting Standby mode, a reset is generated, which resets V_{DDCORE} and V_{DDCPU} logic, excluding the debug logic. Waking up from debug Standby modes, the system CPUs and logic are reset, as from normal Standby mode. SBF, SBF_D1 and SBF_D2 are set.

19.8 PWR WKUP pins

The WKUP polarity must only be changed by configuring WKUPP in PWR_WKUPPCRx when the wake-up interrupt is disabled in WKUPENCPU1/2 bits. To avoid false interrupts, WKUPF must be cleared with WKUPC in PWR_WKUPPCRx, before enabling the wake-up interrupt in WKUPENCPU1/2.

19.9 PWR_ON, PWR_CPU_ON and PWR_LP pins

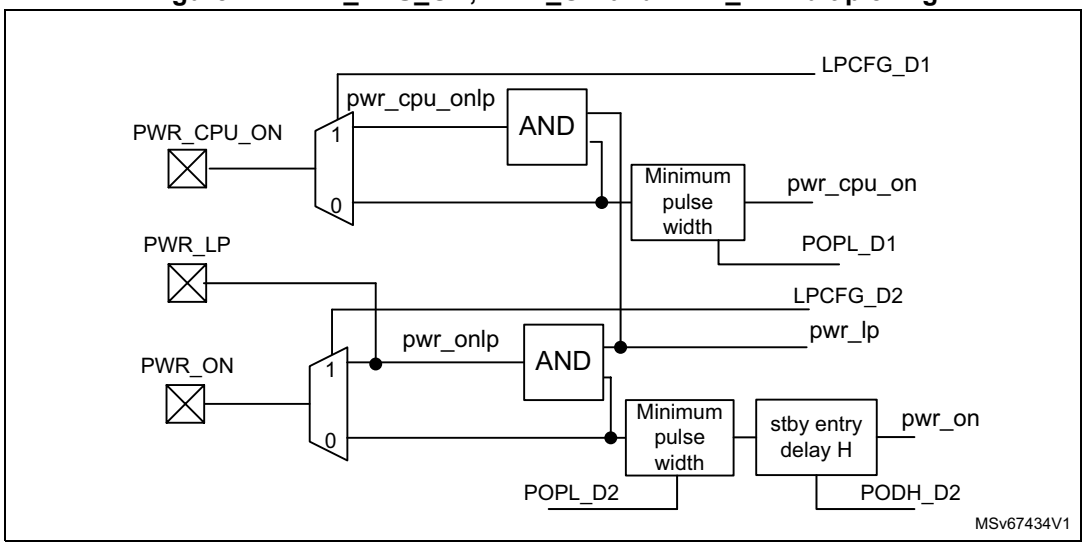
V_{DDCORE} and V_{DDCPU} domains can be independently controlled using PWR_ON, PWR_CPU_ON, and PWR_LP pins.

PWR_CPU_ON signals pwr_cpu_on or pwr_cpu_onlp depending on LPCFG_D1 in PWR_DC1R.

PWR_ON signals pwr_on or pwr_onlp depending on LPCFG_D2 in PWR_D2CR.

PWR_LP signals pwr_onlp.

Figure 72. PWR_CPU_ON, PWR_ON and PWR_LP multiplexing



When LPCFG_D1 selects pwr_cpu_on, PWR_CPU_ON pins are only low in DStandby.

When LPCFG_D1 selects pwr_cpu_onlp on PWR_CPU_ON, the definition (DStandby or DStop2/3) of the signal being low must be communicated to the external power management prior entering low-power mode. PWR_CPU_ON pins are low in DStandby, DStop2, and DStop3 modes.

When LPCFG_D2 selects pwr_on, PWR_ON pin is only low in Standby.

When LPCFG_D2 selects pwr_onlp on PWR_ON, the definition (Standby, LP-Stop1, LP-Stop2, LPLV-Stop1, or LPLV-Stop2) of the signal being low must be communicated to the

external power management prior entering low-power mode. PWR_ON pin is low in Standby, LP-Stop1, LP-Stop2, LPLV-Stop1, and LP-Stop1 modes.

The same external regulator can be used to supply V_{DDCORE} and V_{DDCPU}. Only PWR_ON and PWR_LP signaling are used to control the external regulator. Run2, Stop2, LP-Stop2, and LPLV-Stop2 modes are no more supported.

19.10 PWR interrupts

The PWR provides the following interrupts:

- Wake-up interrupt to CPU1
  - enabled by WKUPENCPU1 in PWR_WKUPPCRx
  - corresponding WKUPF available in PWR_WKUPPCRx
- Wake-up interrupt to CPU2
  - enabled by WKUPENCPU2 in PWR_WKUPPCRx
  - corresponding WKUPF available in PWR_WKUPPCRx

Other status flags associated with Stop, Standby and monitoring functions are not present on all device part numbers (see the related datasheet for more information).

Table 109. PWR status flags

Status flag	Description	Interrupt	Cleared through
STOPF	Exit Stop	Wake-up from Stop events	CSSF
SBF	Exit Standby	Wake-up from Standby events	CSSF
SBF_D1	D1 exits DStandby	Wake-up from DStandby events	CSSF
SBF_D2	D2 exits Standby	Wake-up from Standby events	CSSF
VBF	Exit VBAT	No associated interrupt	CSSF
WKUPF[6:1]	Wake-up flags	CPU1 and CPU2 wake up from Standby and Stop events.	WKUPC[6:1]
PVDO	Programmable voltage detector	Interrupt via EXTI1	NA
GPVMO	Global peripheral voltage monitor	Interrupt via EXTI1	NA
V08CAPL, V08CAPH	V_{08CAP} supply monitoring	Interrupt via EXTI2	NA
TEMPL, TEMPH	Temperature monitoring	Interrupt via EXTI2	NA

19.11 PWR resource isolation framework (RIF)

In order to be compliant with the device security definition, the PWR implements the following security/isolation requirements which apply to any defined security domain:

- filtering of secure/non-secure accesses
- privileged/unprivileged accesses
- filtering of hardware execution compartments (called CIDs): up to eight CIDs including TDCID (trusted domain compartment ID) and DDCID (debug domain compartment ID)

There are three different profiles for PWR local resources:

- single access (only one master can access)
- shared between different masters with semaphore protection
- no access

There are two kinds of local resources in the PWR:

- non-shareable resources (R0 to R6), that can be statically assigned to only one master
- shareable wake-up I/O resources (WIO1 to WIO6) that can be statically assigned to one master, or shared between different masters with semaphore protection

19.11.1 PWR RIF configuration

After any application or system reset, the PWR does not filter any access (default configuration: no CID filtering, non-secure, any privilege) until the TDCID agent has programmed the RIF configuration and enabled the filtering. The PWR RIF configuration is based on the local and external RIF configurations.

The local RIF configuration is managed by the following registers:

- non-shareable resources (R0 to R6)
  - PWR_RSECCFGR to set a secure/non-secure attribute per local PWR resource
  - PWR_RPRIVCFGR to set a privileged/unprivileged attribute per local PWR resource
  - PWR_RxCIDCFGR to set a CID filtering policy per local PWR resource
- shareable wake-up I/O resources (WIO1 to WIO6)
  - PWR_WIOSECCFGR to set a secure/non-secure attribute per local PWR resource
  - PWR_WIOPRIVCFGR to set a privileged/unprivileged attribute per local PWR resource
  - PWR_WIOxCIDCFGR and PWR_WIOxSEMCR to set a CID filtering policy per local PWR resource

The external RIF configuration is inherited from the RIF peripheral configurations and it applies to the related local resources. *Table 110* shows all PWR local resource content (first column) and the RIF configuration attached to it (third column).

Secure/non-secure access filtering

To enable the filtering access based on this attribute, the authorized master agent has to set RSECx in PWR_RSECCFGR, or WIOSECx in PWR_WIOSECCFGR, for the PWR local resource x.

When the local resource is configured as secure, read/write operations are only allowed by a secure access. Non-secure read/write accesses are denied (RAZ/WI). An illegal secure access event is generated to the IAC (illegal access controller). There is no bus error generated.

When the local resource is configured as non-secure, read/write operations are allowed by both secure and non-secure accesses.

Privileged/unprivileged access filtering

To enable the filtering access based on this attribute, the authorized master agent has to set RPRIVx in PWR_RPRIVCFGR, or WIOPRIVx in PWR_WIOPRIVCFGR, for the PWR local resource x.

When the local resource is configured as privileged, read/write operations are only allowed by a privileged access. Unprivileged read/write accesses are denied (RAZ/WI). An illegal privileged access event is generated to the IAC. There is no bus error generated.

When the local resource is configured as unprivileged, read/write operations are allowed by both privileged and unprivileged accesses.

CID access filtering

The PWR_RxCIDCFGR register controls the CID filtering policy for each non-shareable local resources (without semaphore mechanism). PWR_WIOxCIDCFGR and PWR_WIOxSEMCR registers control the CID filtering policy for each local shareable resources (with semaphore mechanism).

When the compartment-based filtering is active (CFEN = 1), local non-shareable resources are individually controlled by a specific CPU. The trusted domain application sets CID in PWR_RxCIDCFGR (x = 0 to 6).

When the compartment-based filtering is active (CFEN = 1), local shareable resources can be individually controlled by a specific CPU using two techniques:

- The trusted domain application keeps SEM_EN cleared, and sets SCID in PWR_WIOxCIDCFGR (x = 1 to 6).
- The trusted domain application sets SEM_EN. It also sets the compartments that are authorized to take the local resource by setting corresponding SEMWLCm in PWR_WIOxCIDCFGR (x = 1 to 6). Each white-listed CPU is authorized to take SEM_MUTEX in PWR_WIOxSEMCR (x = 1 to 6) to have ownership on this register until this CPU releases the mutex.

Any read access is allowed on PWR_RxCIDCFGR, PWR_WIOxCIDCFGR, and PWR_WIOxSEMCR registers.

A non-allowed CID access to a CID-filtered register is denied (RAZ/WI). An illegal CID access event is generated to the IAC. There is no bus error generated.

For debug purpose, the debug domain compartment master, defined by the DDCID compartment value, can access any resource if privileged and secure attributes allow it.

19.12 PWR registers

The PWR registers can be accessed in word, half-word, and byte format, unless explicitly specified differently.

The table below gives the RIF allocation for all PWR registers.

Table 110. PWR register allocation to RIF resource and access conditions

Register name	PWR RIF resource number	RIF resource configuration	Access type(1)	Description
PWR_R0CIDCFGR	R0	Secure, privileged and TDCID/DDCID	RW	Resource 0 CID filtering control
PWR_CR1		PWR_RSECCFGR[0] PWR_RPRIVCFGR[0] PWR_R0CIDCFGR	RW	Peripheral voltage monitor
PWR_CR2			RW	V_{08CAP} and TEMP monitor
PWR_CR3			RW	Programmable voltage detector
PWR_R1CIDCFGR	R1	Secure, privileged and TDCID/DDCID	RW	Resource 1 CID filtering control
PWR_CR9		PWR_RSECCFGR[1] PWR_RPRIVCFGR[1] PWR_R1CIDCFGR	RW	BKPRAM low-power control
PWR_CR10			RW	RETRAM low-power control
PWR_D1CR			RW	V_{DDCPU} power pin control
PWR_D2CR			RW	V_{DDCORE} power pin control
PWR_R2CIDCFGR	R2	Secure, privileged and TDCID/DDCID	RW	Resource 2 CID filtering control
PWR_CPU1CR		PWR_RSECCFGR[2] PWR_RPRIVCFGR[2] PWR_R2CIDCFGR	RW	CPU1 power control
PWR_R3CIDCFGR	R3	Secure, privileged and TDCID/DDCID	RW	Resource 3 CID filtering control
PWR_CPU2CR		PWR_RSECCFGR[3] PWR_RPRIVCFGR[3] PWR_R3CIDCFGR	RW	CPU2 power control
PWR_R4CIDCFGR	R4	Secure, privileged and TDCID/DDCID	RW	Resource 4 CID filtering control
Reserved		PWR_RSECCFGR[4] PWR_RPRIVCFGR[4] PWR_R4CIDCFGR	RW	Reserved
PWR_R5CIDCFGR	R5	Secure, privileged and TDCID/DDCID	RW	Resource 5 CID filtering control
PWR_CR7		PWR_RSECCFGR[5] PWR_RPRIVCFGR[5] PWR_R5CIDCFGR	RW	V_{DDIO2} peripheral voltage monitor
PWR_R6CIDCFGR	R6	Secure, privileged and TDCID/DDCID	RW	Resource 6 CID filtering control
PWR_CR8		PWR_RSECCFGR[6] PWR_RPRIVCFGR[6] PWR_R6CIDCFGR	RW	V_{DDIO1} peripheral voltage monitor

Table 110. PWR register allocation to RIF resource and access conditions (continued)

Register name	PWR RIF resource number	RIF resource configuration	Access type(1)	Description
PWR_WIO1CIDCFGR	WIO1	Secure, privileged and TDCID/DDCID	RW	Wake-up I/O resource 1 CID filtering control
PWR_WIO1SEMCR		PWR_WIOSECCFGR[0] PWR_WIOPRIVCFGR[0] PWR_WIO1CIDCFGR PWR_WIO1SEMCR	RW	
PWR_WKUPCR1			RW	Wake-up pin #1 control
PWR_WIO2CIDCFGR	WIO2	Secure, privileged and TDCID/DDCID	RW	Wake-up I/O resource 2 CID filtering control
PWR_WIO2SEMCR		PWR_WIOSECCFGR[1] PWR_WIOPRIVCFGR[1] PWR_WIO2CIDCFGR PWR_WIO2SEMCR	RW	
PWR_WKUPCR2			RW	Wake-up pin #2 control
PWR_WIO3CIDCFGR	WIO3	Secure, privileged and TDCID/DDCID	RW	Wake-up /IO resource 3 CID filtering control
PWR_WIO3SEMCR		PWR_WIOSECCFGR[2] PWR_WIOPRIVCFGR[2] PWR_WIO3CIDCFGR PWR_WIO3SEMCR	RW	
PWR_WKUPCR3			RW	Wake-up pin #3 control
PWR_WIO4CIDCFGR	WIO4	Secure, privileged and TDCID/DDCID	RW	Wake-up I/O resource 4 CID filtering control
PWR_WIO4SEMCR		PWR_WIOSECCFGR[3] PWR_WIOPRIVCFGR[3] PWR_WIO4CIDCFGR PWR_WIO4SEMCR	RW	
PWR_WKUPCR4			RW	Wake-up pin #4 control
PWR_WIO5CIDCFGR	WIO5	Secure, privileged and TDCID/DDCID	RW	Wake-up I/O resource 5 CID filtering control
PWR_WIO5SEMCR		PWR_WIOSECCFGR[4] PWR_WIOPRIVCFGR[4] PWR_WIO5CIDCFGR PWR_WIO5SEMCR	RW	
PWR_WKUPCR5			RW	Wake-up pin #5 control
PWR_WIO6CIDCFGR	WIO6	Secure, privileged and TDCID/DDCID	RW	Wake-up I/O resource 6 CID filtering control
PWR_WIO6SEMCR		PWR_WIOSECCFGR[5] PWR_WIOPRIVCFGR[5] PWR_WIO6CIDCFGR PWR_WIO6SEMCR	RW	
PWR_WKUPCR6			RW	Wake-up pin #6 control
PWR_CR11	NA	Inherited from RCC internal resource 104 RIF configuration RCC_SECCFGR3[8] RCC_PRIVCFGR3[8] RCC_CIDCFG_LOCKR3[8] RCC_R104CIDCFGR RCC_R104SEMCR	RW	DDR PHY retention control

Table 110. PWR register allocation to RIF resource and access conditions (continued)

Register name	PWR RIF resource number	RIF resource configuration	Access type(1)	Description
PWR_BDCR	NA	Secure, privileged and TDCID/DDCID	RW	Backup domain control register
PWR_CPU1D1SR	NA	NA	R	PWR CPU1 and D1 domain status register
PWR_CPU2D2SR	NA	NA	R	PWR CPU2 and D2 domain status register
PWR_VERR	NA	NA	R	PWR version register
PWR_IPIDR	NA	NA	R	PWR identification register
PWR_SIDR	NA	NA	R	PWR size identification register

1. RW = read or write, R = read, W = write

19.12.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

See individual bits for reset condition. Access 6 wait states when writing this register.

When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC0 in PWR_RSECCFGR. Privileged restriction is defined by RPRIV0 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPVMO	Res.	Res.	Res.	Res.	Res.	VDDIO3VRSEL	VDDIOVRSEL	Res.	Res.	Res.	ARDY	Res.	USB33RDY	Res.	VDDIO3RDY
r						rw	rw				r		r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASV	Res.	Res.	Res.	VDDIO3SV	Res.	Res.	Res.	AVMEN	Res.	USB33VMEN	Res.	VDDIO3VMEN
			rw				rw				rw		rw		rw

Bit 31 **GPVMO:** Global peripheral voltage monitor output

This bit is set and cleared by hardware. It is valid only if at least on independent power supply monitor (including V_{DDIO1} and V_{DDIO2} monitors) is enabled, and the supply is valid (voltage monitor enable bit and supply valid bit must be set for the same power supply).

0: All enabled/valid independent power supplies are equal or higher than PVM threshold.

1: At least one enabled/valid independent power supply is lower than PVM threshold.

Bits 30:26 Reserved, must be kept at reset value.

Bit 25 **VDDIO3VRSEL:** VDDIO3 I/O voltage range selection

This bit is used to select the voltage range supported by the I/Os supplied by V_{DDIO3}.It is not reset by wake-up from Standby, but by any application reset (such as NRST, IWDG).

0: 3.3 V voltage range selected. If V_{DDIO3} is in 1.8 V range with this setting, the I/Os work in degraded mode.

1: 1.8 V voltage range selected. HSLV_VDDIO3 OTP bit must be set to allow 1.8 V voltage range operation. Setting this configuration while V_{DDIO3} is in 3.3 V range damages the device.

Bit 24 **VDDIOVRSEL:** VDD I/O voltage range selection

This bit is used to select the voltage range supported by the I/Os supplied by V_{DD}. It is not reset by wake-up from Standby, but by any application reset (such as NRST, IWDG).

0: 3.3 V voltage range selected. If V_{DD} is in 1.8 V range with this setting, the I/Os work in degraded mode.

1: 1.8 V voltage range selected. HSLV_VDD OTP bit must be set to allow 1.8 V voltage range operation. Setting this configuration while V_{DD} is in 3.3 V range damages the device.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ARDY**: $V_{DDA18ADC}$ ready

0: $V_{DDA18ADC}$ is below the threshold of the $V_{DDA18ADC}$ voltage monitor.

1: $V_{DDA18ADC}$ is equal or above the threshold of the $V_{DDA18ADC}$ voltage monitor.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **USB33RDY**: $V_{DD33USB}$ ready

0: $V_{DD33USB}$ is below the threshold of the USB33 voltage monitor.

1: $V_{DD33USB}$ is equal or above the threshold of the USB33 voltage monitor.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **VDDIO3RDY**: V_{DDIO3} ready

0: V_{DDIO3} is below the threshold of the V_{DDIO3} voltage monitor.

1: V_{DDIO3} is equal or above the threshold of the V_{DDIO3} voltage monitor.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASV**: $V_{DDA18ADC}$ independent supply valid

This bit is used to validate $V_{DDA18ADC}$ supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the A/D converters. If $V_{DDA18ADC}$ is not always present in the application, the $V_{DDA18ADC}$ voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: $V_{DDA18ADC}$ is not present. Logical and electrical isolation is applied to ignore this supply.

1: $V_{DDA18ADC}$ is valid.

Bit 11:9 Reserved, must be kept at reset value.

Bit 8 **VDDIO3SV**: V_{DDIO3} independent supply valid

This bit is used to validate the V_{DDIO3} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PD[11:0] I/Os. If V_{DDIO3} is not always present in the application, the V_{DDIO3} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: V_{DDIO3} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO3} is valid.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **AVMEN:** $V_{DDA18ADC}$ independent ADC voltage monitor enable
This bit is reset on any system reset.
0: $V_{DDA18ADC}$ voltage monitor disabled.
1: $V_{DDA18ADC}$ voltage monitor enabled.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **USB33VMEN:** $V_{DD33USB}$ independent USB 33 voltage monitor enable
This bit is reset on any system reset.
0: $V_{DD33USB}$ voltage monitor disabled.
1: $V_{DD33USB}$ voltage monitor enabled.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **VDDIO3VMEN:** V_{DDIO3} independent voltage monitor enable
This bit is reset on any system reset.
0: V_{DDIO3} voltage monitor disabled.
1: V_{DDIO3} voltage monitor enabled.

19.12.2 PWR control register 2 (PWR_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

Not reset by wake-up from Standby, application reset (such as NRST, IWDG) and V_{DD} POR, but reset only by V_{SW} POR and VSWRST. Access 6 wait states when writing this register.

After reset, this register is write-protected, and DBD in PWR_BDCR has to be set before this register can be written. When DBD is cleared, there are no bus errors generated when writing this register. This register must not be accessed when RCC VSWRST in RCC_BDCR resets the V_{SW} domain. Secure restriction is defined by RSEC0 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV0 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TEMPH	TEMPL	V08CAPH	V08CAPL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN
				r	r	r	r								rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TEMPH**: Monitored temperature level above high threshold
0: Temperature below high threshold level, or monitor disabled
1: Temperature equal or above high threshold level

Bit 10 **TEMPL**: Monitored temperature level above low threshold

0: Temperature above low threshold level, or monitor disabled

1: Temperature equal or below low threshold level

Bit 9 **V08CAPH**: Monitored V_{08CAP} level above high threshold

0: V_{08CAP} level below high threshold level, or monitor disabled

1: V_{08CAP} level equal or above high threshold level

Bit 8 **V08CAPL**: Monitored V_{08CAP} level above low threshold

0: V_{08CAP} level above low threshold level, or monitor disabled

1: V_{08CAP} level equal or below low threshold level

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **MONEN:** V_{08CAP} and temperature monitoring enable

0: V_{08CAP} and temperature monitoring disabled

1: V_{08CAP} and temperature monitoring enabled

19.12.3 PWR control register 3 (PWR_CR3)

Address offset: 0x008

Reset value: 0x0000 0000

Reset on any system reset.

Secure restriction is defined by RSEC0 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV0 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDEN
							r								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PVDO**: Programmable voltage detector (PVD) output

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDEN bit.

0: Voltage level on PVD_IN is equal or higher than the internal V_{REFINT}.

1: Voltage level on PVD_IN is lower than the internal V_{REFINT}.

Note: The PVD is disabled in Standby and after a system reset. This bit is then equal to 0 after Standby and system reset.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **PVDEN:** PVD enable

This bit is read only when the SYSCFG register bit PVDL is set (no bus errors generated when writing this register).

0: PVD disabled
1: PVD enabled

19.12.4 PWR control register 7 (PWR_CR7)

Address offset: 0x018

Reset value: 0x0000 0000

See individual bits for reset condition. Access 6 wait states when writing this register.

When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC5 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV5 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2VRSTBY	VDDIO2VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2RDY
						rw	rw								r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2VMEN
							rw								rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **VDDIO2VRSTBY:** V_{DDIO2} I/O voltage range Standby mode

When set, this bit is used to retain VDDIO2VRSEL configuration in Standby mode.
This bit must be written to 1 if the V_{DDIO2} is in 1.8 V range in Standby, and when exiting Standby. It must not be set when V_{DDIO2} is in 3.3 V range in Standby, or when exiting Standby. This bit is not reset by wake-up from Standby, but by any application reset (such as NRST or IWDG).

0: VDDIO2VRSEL not retained in Standby mode
1: VDDIO2VRSEL retained in Standby mode

Bit 24 **VDDIO2VRSEL:** V_{DDIO2} I/O voltage range selection

This bit is used to select the voltage range supported by the I/Os supplied by V_{DDIO2}. It is not reset by wake-up from Standby, but by any Application reset (such as NRST, IWDG) when VDDIO2VRSTBY is set, otherwise reset on any system reset.

0: 3.3 V voltage range selected. If V_{DDIO2} is in 1.8 V range with this setting, the I/Os work in degraded mode.
1: 1.8 V voltage range selected. HSLV_VDDIO2 OTP bit must be set to allow 1.8 V voltage range operation. Setting this configuration while V_{DDIO2} is in 3.3 V range damages the device.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **VDDIO2RDY**: V_{DDIO2} ready

0: V_{DDIO2} is below the threshold of the V_{DDIO2} voltage monitor.

1: V_{DDIO2} is equal or above the threshold of the V_{DDIO2} voltage monitor.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **VDDIO2SV**: V_{DDIO2} Independent supply valid.

This bit is used to validate the V_{DDIO2} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PE[15:6] I/Os. If V_{DDIO2} is not always present in the application, the V_{DDIO2} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: V_{DDIO2} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO2} is valid.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **VDDIO2VMEN:** V_{DDIO2} independent voltage monitor enable

This bit is reset on any system reset.

0: V_{DDIO2} voltage monitor disabled

1: V_{DDIO2} voltage monitor enabled

19.12.5 PWR control register 8 (PWR_CR8)

Address offset: 0x01C

Reset value: 0x0000 0000

See individual bits for reset condition. Access 6 wait states when writing this register.

When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC6 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV6 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	VDDIO 1VRSTBY	VDDIO 1VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO 1RDY
						rw	rw								r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO 1SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO 1VMEN
							rw								rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **VDDIO1VRSTBY:** V_{DDIO1} I/O voltage range Standby mode

When set, this bit is used to retain VDDIO1VRSEL configuration in Standby. This bit must be written to 1 if V_{DDIO1} is in 1.8 V range in Standby, and when exiting Standby. It must not be set when V_{DDIO1} is in 3.3 V range in Standby, or when exiting Standby. This bit is not reset by wake-up from Standby, but by any Application reset (such as NRST, IWDG).

0: VDDIO1VRSEL not retained in Standby mode

1: VDDIO1VRSEL bit retained in Standby mode

Bit 24 **VDDIO1VRSEL:** V_{DDIO1} I/O voltage range selection

This bit is used to select the voltage range supported by the I/Os supplied by V_{DDIO1}. It is not reset by wake-up from Standby, but by any application reset (such as NRST, IWDG) when VDDIO1VRSTBY is set, otherwise reset on any system reset.

0: 3.3 V voltage range selected. If V_{DDIO1} is in 1.8 V range with this setting, the I/Os work in degraded mode.

1: 1.8 V voltage range selected. HSLV_VDDIO1 OTP bit must be set to allow 1.8 V voltage range operation. Setting this configuration while V_{DDIO1} is in 3.3 V range damages the device.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **VDDIO1RDY**: V_{DDIO1} ready

0: V_{DDIO1} is below the threshold of the V_{DDIO1} voltage monitor.

1: V_{DDIO1} is equal or above the threshold of the V_{DDIO1} voltage monitor.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **VDDIO1SV**: V_{DDIO1} independent I/O supply valid

This bit is used to validate the V_{DDIO1} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PE[5:0] I/Os. If V_{DDIO1} is not always present in the application, the V_{DDIO1} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: V_{DDIO1} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO1} is valid.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **VDDIO1VMEN:** V_{DDOI1} independent I/O voltage monitor enable

This bit is reset on any system reset.

0: V_{DDIO1} voltage monitor disabled

1: V_{DDIO1} voltage monitor enabled

19.12.6 PWR control register 9 (PWR_CR9)

Address offset: 0x020

Reset value: 0x0000 0000

Not reset by wake-up from Standby mode, application reset (such as NRST, IWDG), and V_{DD} POR, but reset only by V_{SW} POR and VSWRST. Access 6 wait states when writing this register. After reset, this register is write-protected, and DBP in PWR_BDCR has to be set before it can be written. When DBP is cleared, there are no bus errors generated when writing this register. This register must not be accessed when RCC VSWRST in RCC_BDCR resets the V_{SW} domain. When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC1 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV1 in PWR_RPRIVCFGR.

If compartment filtering is enabled only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRBSEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BKPRBSEN:** Backup RAM backup supply enable

This bit is used to maintain BKPSRAM content in Standby and V_{BAT} modes.When it is set, the BKPSRAM is supplied from backup regulator in Standby and V_{BAT}. When this bit is reset, the BKPSRAM can still be used in Run and Stop, but its content is lost in Standby and V_{BAT}.

0: BKPSRAM backup supply disabled

1: BKPSRAM backup supply enabled

19.12.7 PWR control register 10 (PWR_CR10)

Address offset: 0x024

Reset value: 0x0000 0000

Not reset by wake-up from Standby, application reset (such as NRST, IWDG), and V_{DD} POR, but reset only by V_{SW} POR and VSWRST. Access 6 wait states when writing this register.

After reset, this register is write-protected, and DBP in PWR_BDCR has to be set before it can be written. When DBP is cleared, there are no bus errors generated when writing this register. This register must not be accessed when RCC VSWRST in RCC_BDCR resets the V_{SW} domain. When a system reset occurs during the register write cycle the written data is not guaranteed. Secure restriction is defined by RSEC1 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV1 in PWR_RPRIVCFGR.

If compartment filtering is enabled only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRBSEN[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **RETRBSEN[1:0]:** Retention RAM backup supply enable

This bit is used to maintain RETRAM content in Standby and V_{BAT} modes. When it is set, the RETRAM is supplied from backup regulator in Standby and V_{BAT}. When this bit is reset, the RETRAM can still be used in Run and Stop, but its content is lost in Standby and VBAT.

00: RETRAM backup supply disabled

01: RETRAM backup supply enabled in Standby and VBAT

10-11: RETRAM backup supply enabled only in Standby (memory content lost in V_{BAT} mode)

Others: Reserved

19.12.8 PWR control register 11 (PWR_CR11)

Address offset: 0x028

Reset value: 0x0000 0000

Reset on any system reset.

Compartment filtering, secure and privilege restriction inherited from RCC internal resource 104 RIF configuration (see *Section 20: Reset and clock control (RCC)* for details. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRRETDIS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DDRRETDIS:** DDR retention disable

When written 1 to this bit, the DDR PHY exits LP3/IO retention. To enable retention software must write this bit to 0.

0: DDR self-refresh retention enabled
1: DDR self-refresh retention disabled

19.12.9 PWR backup domain control register (PWR_BDCR)

Address offset: 0x038

Reset value: 0x0000 0000

Reset on any system reset.

Secure privileged write access only. Write only by trusted domain CID, as defined in RIMC. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBP*:*** Disable backup write protection

In reset state, all registers in backup domain are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Write access to backup domain disabled
1: Write access to backup domain enabled

19.12.10 PWR CPU1 control register (PWR_CPU1CR)

Address offset: 0x040

Reset value: 0x0000 0000

See individual bits for reset condition. Access 6 wait states when writing this register. When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC2 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV2 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVDS_D1	LPDS_D1
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STANDBYWFIL2	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBF_D1	SBF	STOPF	VBF	Res.	Res.	PDDS_D1	PDDS_D2
r						w		r	r	r	rw			rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LVDS_D1:** Low-voltage Deepsleep LPLV-Stop mode selection for the D1 domain (DStop3)
This bit is reset on any system reset. It has effect only when LPDS_D1 = 1. It changes the V_{DDCPU} domain supply reset level.
0: LP-Stop mode V_{DDCPU} domain supply reset level at same level as Run mode.
V_{DDCPU} domain supply level in LP-Stop mode must be kept at same level as Run mode.
1: LPLV-Stop mode V_{DDCPU} domain supply reset level at lower level than Run mode.
V_{DDCPU} supply can be lowered in LPLV-Stop mode (see datasheet for voltage levels).

Bit 16 **LPDS_D1:** low-power Deepsleep Stop mode selection for the D1 domain (DStop2)
This bit is reset on any system reset.
0: Stop mode selected, external regulator kept in main power mode.
1: LP/LPLV-Stop mode selected, external regulator may enter low-power mode (low-power mode selection provided by LVDS_D1).

Bit 15 **STANDBYWFIL2:** CPU1 system idle indication
This bit is set and reset by hardware based on the CPU1 operation mode.
0: CPU1 system active
1: CPU1 system idle

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **CSSF:** Clear CPU1 STOPF, SBF_D1, SBF, VBF and CPU2 SBF flags (Always read as 0)
This bit is reset on any system reset.
0: No effect.
1: Clear CPU1 (STOPF, SBF_D1, SBF, VBF) and CPU2 SBF flags. Register bit is cleared to 0 by hardware.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SBF_D1:** D1 DStandby flag

This bit is set by hardware and cleared only by a V_{DD} POR reset, or by setting the CPU1 CSSF bit. It is not reset when exiting Standby mode.

0: D1 domain has not been in DStandby mode.

1: D1 domain has been in DStandby mode.

Bit 6 **SBF:** System Standby flag

This bit is set by hardware and cleared only by a V_{DD} POR reset, or by setting the CPU1 or CPU2 CSSF bit. It is not reset when exiting Standby mode.

0: System has not been in Standby mode.

1: System has been in Standby mode and system contents has been lost.

Bit 5 **STOPF:** System Stop flag

This bit is set by hardware, and cleared only by any system reset, or by setting the CPU1 CSSF bit.

0: System has not been in Stop mode.

1: System has been in Stop mode, and the clock system has been stopped.

Bit 4 **VBF:** V_{BAT} mode exit flag

This bit is set by hardware, and cleared only by a V_{sw} POR reset, or by setting the CPU1 CSSF bit. It is not reset when exiting V_{BAT} mode.

0: System has not been in V_{BAT} mode.

1: System has been in V_{BAT} mode.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **PDDS_D1:** D1 domain power-down Deepsleep selection

This bit is reset only by a V_{DD} POR reset. It is not reset when exiting Standby. It allows CPU1 to define the Deepsleep mode for the system.

0: DStop mode kept when CPU1 enters CStop

1: DStandby mode allowed when CPU1 enters CStop

Bit 0 **PDDS_D2*:*** D2 domain/system power-down Deepsleep selection

This bit is reset only by a V_{DD} POR reset. It is not reset when exiting Standby. it allows CPU1 to define the Deepsleep mode for the D2/system.

0: Stop mode kept when CPU1 enters CStop

1: Standby mode allowed when CPU1 enters CStop

19.12.11 PWR CPU2 control register (PWR_CPU2CR)

Address offset: 0x044

Reset value: 0x0000 0000

See individual bits for reset condition. Access 6 wait states when writing this register. When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC3 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV3 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVDS_D2	LPDS_D2
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEEPSLEEP	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBF_D2	SBF	STOPF	VBF	Res.	Res.	Res.	PDDS_D2
r						w		r	r	r	rw				rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LVDS_D2:** Low-voltage Deepsleep LPLV-Stop mode selection for the D2 domain/system
This bit is reset on any system reset. It has effect only when LPDS_D2 = 1. It changes the V_{DDCORE} domain supply reset level.
0: LP-Stop mode V_{DDCORE} domain supply reset level at same level as Run mode. V_{DDCORE} domain supply level in LP-Stop mode must be kept at same level as Run mode.
1: LPLV-Stop mode V_{DDCORE} domain supply reset level at lower level than Run mode. V_{DDCORE} supply can be lowered in LPLV-Stop mode (see datasheet for voltage levels).

Bit 16 **LPDS_D2:** low-power Deepsleep Stop mode selection for the D2 domain/syste.
This bit is reset on any system reset.
0: Stop mode selected, external regulator kept in main power mode (pwr_lp = 1)
1: LP/LPLV-Stop mode selected, external regulator may enter low-power mode (pwr_lp = 0) (low-power mode selection provided by LVDS_D2).

Bit 15 **DEEPSLEEP:** CPU2 system idle indication
This bit is set and reset by hardware based on the CPU2 operation mode.
0: CPU2 system active
1: CPU2 system idle

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **CSSF:** Clear CPU2 STOPF, SBF_D2, SBF, VBF and CPU1 SBF flags (Always read as 0)
This bit is reset on any system reset.
0: No effect
1: Clear CPU2 (STOPF, SBF_D2, SBF, VBF) and CPU1 SBF flags. Register bit is cleared to 0 by hardware.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SBF_D2:** D2 Standby flag

This bit is set by hardware, and cleared only by a V_{DD} POR reset, or by setting the CPU2 CSSF bit. It is not reset when exiting Standby mode.

0: D2 domain has not been in Standby mode.

1: D2 domain has been in Standby mode.

Bit 6 **SBF:** System Standby flag

This bit is set by hardware, and cleared only by a V_{DD} POR reset, or by setting the CPU2 or CPU1 CSSF bit. It is not reset when exiting Standby mode.

0: System has not been in Standby mode

1: System has been in Standby mode, system contents has been lost.

Bit 5 **STOPF:** System Stop flag

This bit is set by hardware, and cleared only by any system reset ,or by setting the CPU2 CSSF bit.

0: System has not been in Stop mode.

1: System has been in Stop mode, and the clock system has been stopped.

Bit 4 **VBF:** V_{BAT} exit flag

This bit is set by hardware, and cleared only by a V_{sw} POR reset, or by setting the CPU2 CSSF bit. i ti snot reset when exiting V_{BAT} mode.

0: System has not been in V_{BAT} mode.

1: System has been in V_{BAT} mode.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDDS_D2*:*** D2 domain/system power-down Deepsleep selection

This bit is reset only by a V_{DD} POR reset. It is not reset when exiting Standby mode. It allows CPU2 to define the Deepsleep mode for the D2/system.

0: Stop mode kept when CPU2 enters CStop

1: Standby mode allowed when CPU2 enters CStop

19.12.12 PWR D1 control register (PWR_D1CR)

Address offset: 0x04C

Reset value: 0x0000 0000

Not reset by wake-up from Standby and application reset (such as NRST, IWDG) but only reset by V_{DD} POR. Access 6 wait states when writing this register. When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC1 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV1 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	POPL_D1[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPCFG_D1
			rw	rw	rw	rw	rw								rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **POPL_D1[4:0]**: pwr_cpu_on pulse low configuration

These bits are set and cleared by software. They define the minimum guaranteed duration of the pwr_cpu_on low pulse in DStandby mode (no impact on DSTOP2/3 modes). The LSI oscillator is automatically enabled when needed by the POPL_D1 pulse low configuration.

00000: No guaranteed minimum low time

00001: ~1 ms guaranteed minimum low time (1 x 32 LSI cycles)

00010: ~2 ms guaranteed minimum low time (2 x 32 LSI cycles)

...

11111: ~31 ms guaranteed minimum low time. (31 x 32 LSI cycles)

Note: POPL_D1 value must not be smaller than PODH_D2 value.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LPCFG_D1:** PWR_CPU_ON pin configuration

0: PWR_CPU_ON pin signals DStandby mode (PWR_CPU_ON = 1 in DRun, DStop1/2/3, and = 0 in DStandby) (pwr_cpu_on).

1: PWR_CPU_ON pin signals DStandby and DStop2/3 modes (PWR_CPU_ON = 1 in DRun, DStop1, and = 0 in DStop2/3 and DStandby) (pwr_cpu_onlp).

19.12.13 PWR D2 control register (PWR_D2CR)

Address offset: 0x050

Reset value: 0x0007 0000

See individual bits for reset condition. Access 6 wait states when writing this register. When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by RSEC1 in PWR_RSECCFGR. Privilege restriction is defined by RPRIV1 in PWR_RPRIVCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PODH_D2[3:0]				Res.	Res.	Res.	Res.	Res.	LPLVDLY_D2[2:0]		
				rw	rw	rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	POPL_D2[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPCFG_D2
			rw	rw	rw	rw	rw								rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **PODH_D2[3:0]**: pwr_on delay high configuration

These bits are set and cleared by software. They define the delay to maintain PWR_ON high when entering in Standby mode. This delay is used to switch off V_{DDCPU} before V_{DDCORE}. There is no impact when exiting Standby or on LP-Stop and LPLV-Stop modes. The LSI oscillator is automatically enabled when needed by the PODH_D2 delay high configuration. These bits are not reset by wake-up from Standby and application reset (such as NRST, IWDG) but only reset by V_{DD} POR.

0000: No delay

0001: ~1 ms delay (1 x 32 LSI cycles)

0010: ~2 ms delay (2 x 32 LSI cycles)

...

1111: ~15 ms delay (15 x 32 LSI cycles)

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **LPLVDLY_D2**[2:0]: LPLV delay value

These bits are set and cleared by software. They define the minimum delay value to be observed before activating PLLs and interconnect clocks, after one of the LPLV system Stop. The LSI oscillator is automatically enabled when needed by the counter. These bits are reset on any system reset.

000: ~187 µs delay (1 x 6 LSI cycles)

001: ~375 µs delay (2 x 6 LSI cycles)

010: ~562 µs delay (3 x 6 LSI cycles)

...

111: ~1.5 ms delay (8 x 6 LSI cycles) (default)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **POPL_D2[4:0]**: pwr_on pulse low configuration

These bits are set and cleared by software. They define the minimum guaranteed duration of the pwr_on low pulse in Standby (no impact on LP-Stop and LPLV-Stop modes). The LSI oscillator is automatically enabled when needed by the POPL_D2 pulse low configuration. These bits are not reset by wake-up from Standby and application reset (such as NRST, IWDG) but only reset by V_{DD} POR.

00000: No guaranteed minimum low time

00001: ~1 ms guaranteed minimum low time (1 x 32 LSI cycles)

00010: ~2 ms guaranteed minimum low time (2 x 32 LSI cycles)

...

11111: ~31 ms guaranteed minimum low time (31 x 32 LSI cycles)

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LPCFG_D2:** PWR_ON pin configuration

This bit is not reset by wake-up from Standby and application reset (such as NRST, IWDG) but only reset by V_{DD} POR.

0: PWR_ON pin signals Standby (PWR_ON = 1 in Run, Stop1/2, LP-Stop1/2, LPLV-Stop1/2, and = 0 in Standby) (pwr_on).

1: PWR_ON pin signals Standby, LP-Stop, and LPLV-Stop modes (PWR_ON = 1 in Run, Stop1/2, and = 0 in LP-Stop1/2, LPLV-Stop1/2 and Standby) (pwr_onlp).

19.12.14 PWR wake-up control register x (PWR_WKUPCRx)

Address offset: 0x060 + 0x4 * (x - 1) (x = 1 to 6)

Reset value: 0x0000 0000

Not reset by wake-up from Standby, but by any application reset (such as NRST, IWDG). Access 6 wait states when writing this register. When clearing a WKUPF, the AHB write access completes after WKUPF has been cleared). When a system reset occurs during the register write cycle, the written data is not guaranteed. Secure restriction is defined by WIOSECx in PWR_WIOSECCFGR. Privilege restriction is defined by WIOPRIVx in PWR_WIOSECCFGR.

If compartment filtering is enabled, only an authorized compartment can write this register, on top of secure and privileged restrictions. Authorized compartment value depends on the value of SEM_EN in PWR_WIOxCIDCFGR, and optionally SEM_MUTEX in PWR_WIOxSEMCR. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPENCPU2	WKUPENCPU1
r														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WKUPPUPD[1:0]		Res.	Res.	Res.	WKUPP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC
		rw	rw				rw								w

Bit 31 **WKUPF:** Wake-up flag for WKUPx pin before enable

This bit is set by hardware and cleared only by a NRST reset, or by setting WKUPC in PWR_WKUPCRx.

0: No wake-up event occurred.

1: A wake-up event was received from WKUPx pin.

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **WKUPENCPU2:** Enable WKUPx pin and interrupt for CPU2

0: An event on WKUPx pin does not wake up the system from Standby or Stop, nor generates an interrupt to CPU2.

1: A rising or falling edge on WKUPx pin wakes up the system from Standby or Stop, and generates an interrupt to CPU2.

Bit 16 **WKUPENCPU1:** Enable WKUPx pin and interrupt for CPU1

0: An event on WKUPx pin does not wake up the system from Standby or Stop, nor generates an interrupt to CPU1.

1: A rising or falling edge on WKUPx pin wakes up the system from Standby or Stop, and generates an interrupt to CPU1.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **WKUPPUPD[1:0]**: Wake-up pull configuration for WKUPx pin
These bits define the I/O pin pull configuration used when WKUPENCPU1 or/and WKUPENCPU2 are set to 1 (associated GPIO port pull configuration must be set to the same value or 00). The wake-up pin pull configuration is maintained in Standby mode.
00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **WKUPP**: Wake-up polarity bit for WKUPx pin
This bit defines the polarity used for event detection on external WKUPx pin.
0: High-level detection (rising edge)
1: Low-level detection (falling edge)

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **WKUPC:** Clear wake-up flag for WKUPx pin
This bit is always read as 0. It is cleared to 0 by hardware.
0: No effect
1: Clear WKUPF in PWR_WKUPCRx

19.12.15 PWR resource secure configuration register (PWR_RSECCFGR)

Address offset: 0x100

Reset value: 0x0000 0000

Reset on any system reset.

Non-secure or unprivileged writes to this register are ignored, while any read access from CPU1 and CPU2 is allowed.

Debug domain can write to this register if the feature is activated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSEC6	RSEC5	RSEC4	RSEC3	RSEC2	RSEC1	RSEC0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **RSEC6**: Secure attribute reference for the local resource number 6
See RSEC0 for details.

Bit 5 **RSEC5**: Secure attribute reference for the local resource number 5
See RSEC0 for details.

Bit 4 **RSEC4**: Secure attribute reference for the local resource number 4
See RSEC0 for details.

Bit 3 **RSEC3**: Secure attribute reference for the local resource number 3
See RSEC0 for details.

Bit 2 **RSEC2**: Secure attribute reference for the local resource number 2
See RSEC0 for details.

Bit 1 **RSEC1**: Secure attribute reference for the local resource number 1
See RSEC0 for details.

Bit 0 **RSEC0**: Secure attribute reference for the local resource number 0
This bit is written by software to enable the secure mode for the local resource 0.
If compartment filtering for this local resource is enabled (resp. disabled) by CFEN in PWR_R0CIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if privilege and security of the access is sufficient.
0: The local resource number 0 is non-secure. Secure access is also possible.
1: The local resource number 0 is secure only.

19.12.16 PWR resource privileged configuration register (PWR_RPRIVCFGR)

Address offset: 0x104

Reset value: 0x0000 0000

Reset on any system reset.

Unprivileged writes to this register are ignored while any read access from CPU1 and CPU2 is allowed.

Debug domain can write to this register if the feature is activated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPRIV6	RPRIV5	RPRIV4	RPRIV3	RPRIV2	RPRIV1	RPRIV0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **RPRIV6:** Privileged attribute reference for the local resource number 6
See RPRIV0 for details.

Bit 5 **RPRIV5:** Privileged attribute reference for the local resource number 5
See RPRIV0 for details.

Bit 4 **RPRIV4:** Privileged attribute reference for the local resource number 4)
See RPRIV0 for details.

Bit 3 **RPRIV3:** Privileged attribute reference for the local resource number 3
See RPRIV0 for details.

Bit 2 **RPRIV2:** Privileged attribute reference for the local resource number 2
See RPRIV0 for details.

Bit 1 **RPRIV1:** Privileged attribute reference for the local resource number 1
See RPRIV0 for details.

Bit 0 **RPRIV0**: Privileged attribute reference for the local resource number 0

This bit is written by software to enable the privileged mode for the local resource 0. If corresponding SEC0 is set in PWR_RSECCFGR, this bit can only be written by a secure privileged application. If compartment filtering for this local resource is enabled (resp. disabled) by CFEN in PWR_R0CIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if privilege and security of the access is sufficient.

0: The local resource number 0is unprivileged. Secure access is also possible.

1: The local resource number 0is privileged only.

19.12.17 PWR resource x CID configuration register (PWR_RxCIDCFGR)

Address offset: 0x108 + 0x4 * x, (x = 0 to 6)

Reset value: 0x0000 0000

Reset on any system reset.

This register is used to configure the local resource x as belonging to a specific compartment (or not). Secure privileged write access only. Write only by trusted domain and secure privileged debug domain CID. Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SCID[2:0]**: Static compartment identifier

When CFEN = 1, this field is used by the trusted domain application to define which compartment has access to the local resource number x.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** Compartment ID filter enable

0: Compartment-based filtering is disabled for resource number x.

1: Compartment-based filtering is enabled for resource number x.

19.12.18 PWR WIO secure configuration register (PWR_WIOSECCFGR)

Address offset: 0x180

Reset value: 0x0000 0000

Reset on any system reset.

Non-secure or unprivileged writes to this register are ignored, while any read access from CPU1 and CPU2 is allowed.

Debug domain can write to this register if the feature is activated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIOSEC6	WIOSEC5	WIOSEC4	WIOSEC3	WIOSEC2	WIOSEC1
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **WIOSEC6**: Secure attribute reference for the wake-up I/O 6
See WIOSEC1 for details.

Bit 4 **WIOSEC5**: Secure attribute reference for the wake-up I/O 5
See WIOSEC1 for details.

Bit 3 **WIOSEC4**: Secure attribute reference for the wake-up I/O 4
See WIOSEC1 for details.

Bit 2 **WIOSEC3**: Secure attribute reference for the wake-up I/O 3
See WIOSEC1 for details.

Bit 1 **WIOSEC2**: Secure attribute reference for the wake-up I/O 2)
See WIOSEC1 for details.

Bit 0 **WIOSEC1**: Secure attribute reference for the wake-up I/O 1
This bit is written by software to enabled the secure mode for the wake-up I/O 1.
If compartment filtering for this wake-up I/O is enabled (resp. disabled) by CFEN bit in PWR_WIO1CIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if privilege and security of the access is sufficient.
Authorized compartment is defined as SCID if SEM_EN=0 in PWR_WIO1CIDCFGR.
if SEM_EN = 1, it is equal to SEMCID, if SEM_MUTEX = 1 in PWR_WIO1SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.
0: The wake-up I/O 1 is non-secure. Secure access is also possible.
1: The wake-up I/O 1 is secure only.

19.12.19 PWR WIO privileged configuration register (PWR_WIOPRIVCFGR)

Address offset: 0x184

Reset value: 0x0000 0000

Reset on any system reset.

Unprivileged writes to this register are ignored while any read access from CPU1 and CPU2 is allowed.

Debug domain can write to this register if the feature is activated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIOPRIV6	WIOPRIV5	WIOPRIV4	WIOPRIV3	WIOPRIV2	WIOPRIV1
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **WIOPRIV6:** Privileged attribute reference for the wake-up I/O 6
See WIOPRIV1 for details.

Bit 4 **WIOPRIV5:** Privileged attribute reference for the wake-up I/O 5
See WIOPRIV1 for details.

Bit 3 **WIOPRIV4:** Privileged attribute reference for the wake-up I/O 4
See WIOPRIV1 for details.

Bit 2 **WIOPRIV3:** Privileged attribute reference for the wake-up I/O 3
See WIOPRIV1 for details.

Bit 1 **WIOPRIV2:** Privileged attribute reference for the wake-up I/O 2
See WIOPRIV1 for details.

Bit 0 **WIOPRIV1**: Privileged attribute reference for the wake-up I/O 1
This bit is written by software to enabled the privileged mode for the wake-up I/O 1.If corresponding SEC1 bit is set in PWR_WIOSECCFGR this bit can only be written by a secure privileged application. If compartment filtering for this wake-up I/O is enabled (resp. disabled) by CFEN in PWR_WIO1CIDCFGR, only an authorized privileged compartment (resp. any compartment) can write this bit, if privileged and security of the access is sufficient.
Authorized compartment is defined as SCID if SEM_EN = 0 in PWR_WIO1CIDCFGR.
If SEM_EN = 1, it is equal to SEMCID, if SEM_MUTEX = 1 in PWR_WIO1SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.
0: The wake-up I/O 1 is unprivileged. Privileged access is also possible.
1: The wake-up I/O 1 is privileged only.

19.12.20 PWR WIO x CID configuration register (PWR_WIOxCIDCFGR)

Address offset: 0x188 + 0x8 * (x - 1) (x = 1 to 6)

Reset value: 0x0000 0000

Reset on any system reset.

This register is used to configure the wake-up I/O x as belonging to a specific compartment (or not). Secure privileged write access only. Write only by trusted domain CID.

Debug domain can also access if the feature is activated. Any read access from CPU1 and CPU2 is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
									rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **SEMWLC7**: Semaphore CID 7 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 7 agent disabled to take the semaphore

1: CID 7 agent enabled to take the semaphore

Bit 22 **SEMWLC6**: Semaphore CID 6 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 6 agent disabled to take the semaphore

1: CID 6 agent enabled to take the semaphore

Bit 21 **SEMWLC5**: Semaphore CID 5 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 5 agent disabled to take the semaphore

1: CID 5 agent enabled to take the semaphore

Bit 20 **SEMWLC4**: Semaphore CID 4 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 4 agent disabled to take the semaphore

1: CID 4 agent enabled to take the semaphore

Bit 19 **SEMWLC3**: Semaphore CID 3 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 3 agent disabled to take the semaphore

1: CID 3 agent enabled to take the semaphore

Bit 18 **SEMWLC2**: Semaphore CID 2 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 2 agent disabled to take the semaphore

1: CID 2 agent enabled to take the semaphore

Bit 17 **SEMWLC1**: Semaphore CID 1 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 1 agent disabled to take the semaphore

1: CID 1 agent enabled to take the semaphore

Bit 16 **SEMWLC0**: Semaphore CID 0 enable

This bit is not ignored when semaphore is disabled (SEM_EN = 0) or compartment filtering is off (CFEN = 0).

0: CID 0 agent disabled to take the semaphore

1: CID 0 agent enabled to take the semaphore

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **SCID[2:0]**: Static CID

Trusted domain application uses this bitfield to statically define which compartment has access to the peripheral x when SEM_EN = 0.

This CID is ignored when semaphore is enabled (SEM_EN = 1) or compartment filtering is off (CFEN = 0).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEM_EN**: Semaphore enable

This bit is ignored if filtering is off (CFEN = 0). When this bit is cleared from 1 by trusted domain application, SEM_MUTEX is cleared to zero.

0: Wake-up I/O x access granted only if CID of the access equals to SCID.

1: Wake-up I/O x access granted only if SEM_MUTEX = 1 and CID of the access equals to SEMCID in PWR_WIOxSEMCR.

Bit 0 **CFEN:** CID filtering enable

Set and cleared by software. For any CFEN value, access controls to wake-up I/O x defined in PWR_WIOPRIVCFGR and PWR_WIOSECCFGR always apply. When this bit is cleared from 1 by trusted domain application, SEM_MUTEX is cleared to zero.

0: Compartment-based filtering defined in PWR_WIOxCIDCFGR and optionally PWR_WIOxSEMCR is disabled for wake-up I/O x.

1: Compartment-based filtering defined in PWR_WIOxCIDCFGR and optionally PWR_WIOxSEMCR is enabled for wake-up I/O x.

19.12.21 PWR WIO x semaphore control register (PWR_WIOxSEMCR)

Address offset: 0x18C + 0x8 * (x - 1) (x = 1 to 6)

Reset value: 0x0000 0000

Reset on any system reset.

If compartment filtering is disabled (CFEN = 0) or semaphore is disabled (SEM_EN = 0) in PWR_WIOxCIDCFGR, any write to this register is ignored. Privileged-only write if WIOPRIVx is set in PWR_WIOPRIVCFGR. Secure-only write if WIOSECx is set in PWR_WIOSECCFGR.

If compartment filtering is enabled and semaphore is enabled (CFEN and SEM_EN set in PWR_WIOxCIDCFGR), only an authorized compartment can write to this register. For this register, authorized compartment is equal to SEMCID if SEM_MUTEX = 1, else it is defined by the CID white list (SEMWLCy) in PWR_WIOxCIDCFGR. There is no authorized compartment if semaphore is disabled (SEM_EN = 0). Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
									r	r	r				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SEMCID[2:0]**: Semaphore current CID

When SEM_EN and CFEN are set in PWR_WIOxCIDCFGR, this read-only bitfield is updated with the CID value of the latest successful acquisition of PWR_WIOxSEMCR semaphore (SEM_MUTEX bit set to 1). In all other cases, SEMCID stays unchanged.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX:** Semaphore mutex

White listed application can set semaphore bit to get exclusive access to wake-up I/O x when SEM_EN is set in PWR_WIOxCIDCFGR (writes are ignored otherwise).
When SEM_MUTEX = 0, only applications belonging to white-listed compartments (defined with SEMWLCy in PWR_WIOxCIDCFGR) can acquire this semaphore.
When SEM_MUTEX = 1, only applications belonging to compartment SEMCID can release this semaphore. This bit is cleared when CFEN or SEM_EN are cleared from 1 by trusted domain application in PWR_WIOxCIDCFGR.
Read 0: Semaphore is free to take.
Read 1: Semaphore is taken.
Write 0: Release semaphore.
Write 1: Acquire semaphore.

19.12.22 PWR CPU1 status register (PWR_CPU1D1SR)

Address offset: 0x200

Reset value: 0x0000 0000

Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DSTATE[2:0]			Res.	Res.	Res.	Res.	CSTATE[1:0]		Res.	HOLD_ BOOT
					r	r	r					r	r		r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **DSTATE[2:0]**: D1 domain state status

Set by hardware to indicate when the D1 domain is in DRun, DStop1/2/3 or DStandby mode.
000: DRun
001: DStop1
010: DStop2
011: DStop3
100: DStandby

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CSTATE[1:0]**: CPU1 cluster state status

Set by hardware to indicate when the CPU1 is in CRun, CSleep or CStop state.
00: CPU1 in reset
01: CPU1 in CRun
10: CPU1 in CSleep
11: CPU1 in CStop

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HOLD_BOOT**: CPU1 HOLD_BOOT status flag.

This flag is not valid when the D1 domain is in DStandby mode. It is set by hardware to indicate when the CPU1 is in HOLD_BOOT state.
0: CPU1 allowed to boot
1: CPU1 maintained in HOLD_BOOT state by hardware

Note: Refer to Section 20 for details on HOLD_BOOT processor state.

19.12.23 PWR CPU2 status register (PWR_CPU2D2SR)

Address offset: 0x204

Reset value: 0x0000 0000

Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DSTATE[2:0]			Res.	Res.	Res.	Res.	CSTATE[1:0]		WFBEN	HOLD_BOOT
					r	r	r					r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **DSTATE[2:0]**: System/D2 state status

Set by hardware to indicate when the system/D2 domain is in Run, Stop, LP-Stop, LPLV-Stop, Standby or V_{BAT} mode.

000: Run

001: Stop

010: LP-Stop

011: LPLV-Stop

100: Standby

101: VBAT

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CSTATE[1:0]**: CPU2 cluster state status flag

Set by hardware to indicate when the CPU2 is in CRun, CSleep or CStop state.

00: CPU2 in reset

01: CPU2 in CRun

10: CPU2 in CSleep

11: CPU2 in CStop

Bit 1 **WFBEN**: CPU2 wait-for-boot enable status flag

Set by hardware to indicate when the CPU2 is in wait-for-boot enable state. Only applicable in wake-up event mode. This flag is not valid when the system is in Standby mode.

0: CPU2 is not in wait-for-boot enable state (bit set at 0 in legacy boot mode).

1: CPU2 is in wait-for-boot enable state. It starts once an event assign to it is raised.

Note: Refer to Section 20 for details on HOLD_BOOT processor state.

Bit 0 **HOLD_BOOT**: CPU2 HOLD_BOOT status flag

This flag is not valid when the system is in Standby. Set by hardware to indicate when the CPU2 is in HOLD_BOOT state.

0: CPU2 is allowed to boot.

1: CPU2 is maintained in HOLD_BOOT state by hardware.

Note: Refer to Section 20 for details on HOLD_BOOT processor state.

19.12.24 PWR VER register (PWR_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision number

Bits 3:0 **MINREV[3:0]:** Minor revision number

19.12.25 PWR identifier register (PWR_IPIDR)

Address offset: 0x3F8

Reset value: 0x0001 0003

Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier of the PWR

19.12.26 PWR size identification register (PWR_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

Any read access from CPU1 and CPU2 is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Decoding space is 1 Kbyte.

19.12.27 PWR register map

Table 111. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR1**	GPVMO	Res.	Res.	Res.	Res.	Res.	VDDIO3VRSEL	VDDIOVRSEL	Res.	Res.	Res.	ARDY	Res.	USB33RDY	Res.	VDDIO3RDY	Res.	Res.	Res.	ASV	Res.	Res.	Res.	VDDIO3SV	Res.	Res.	Res.	AVMEN	Res.	USB33VMEN	Res.	VDDIO3VMEN
	Reset value	0						0	0				0		0		0				0				0				0		0		0
0x004	**PWR_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	V08CAPH	V08CAPL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN
	Reset value																					0	0	0	0								0
0x008	**PWR_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDEN
	Reset value																								0								0
0x00C	Reserved	Reserved																															
0x018	**PWR_CR7**	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2VRSTBY	VDDIO2VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO2VMEN
	Reset value							0	0								0								0								0
0x01C	**PWR_CR8**	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO1VRSTBY	VDDIO1VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO1SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO1VMEN
	Reset value							0	0								0								0								0
0x020	**PWR_CR9**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRBSEN
	Reset value																																0
0x024	**PWR_CR10**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRBSEN [1:0]	
	Reset value																															0	0
0x028	**PWR_CR11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRRETDIS
	Reset value																																0

Table 111. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C-0x034	Reserved	Reserved																															
0x038	**PWR_BDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x03C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x040	**PWR_CPU1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVDS_D1	LPDS_D1	STANDBYWFIL2	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBF_D1	SBF	STOPF	VBF	Res.	Res.	PDDS_D1	PDDS_D2
	Reset value															0	0	0						0		0	0	0	0			0	0
0x044	**PWR_CPU2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVDS_D2	LPDS_D2	DEEPSLEEP	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBF_D2	SBF	STOPF	VBF	Res.	Res.	Res.	PDDS_D2
	Reset value															0	0	0						0		0	0	0	0				0
0x048	Reserved																																
0x04C	**PWR_D1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POPL_D1[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPCFG_D1
	Reset value																				0	0	0	0	0								0
0x050	**PWR_D2CR**	Res.	Res.	Res.	Res.	PODH_D2 [3:0]				Res.	Res.	Res.	Res.	Res.	LPLVDLY_D2 [2:0]			Res.	Res.	Res.	POPL_D2[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPCFG_D2
	Reset value					0	0	0	0						1	1	1				0	0	0	0	0								0
0x054-0x05C	Reserved	Reserved																															
0x060 + 0x4*(x-1) (x=1 to 6) Last address: 0x074	**PWR_WKUPCRx**	WKUPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPENCPU2	WKUPENCPU1	Res.	Res.	WKUPPUPD [1:0]					WKUPP								WKUPC
	Reset value	0														0	0			0	0				0								0
0x078-0x0FC	Reserved	Reserved																															
0x100	**PWR_ RSECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSEC6	RSEC5	RSEC4	RSEC3	RSEC2	RSEC1	RSEC0
	Reset value																										0	0	0	0	0	0	0
0x104	**PWR_ RPRIVCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPRIV6	RPRIV5	RPRIV4	RPRIV3	RPRIV2	RPRIV1	RPRIV0
	Reset value																										0	0	0	0	0	0	0

Table 111. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x108 + 0x4*x (x=0 to 6) Last address: 0x120	**PWR_ RxCIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID [2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x124-0x17C	Reserved	Reserved																															
0x180	**PWR_ WIOSECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIOSEC6	WIOSEC5	WIOSEC4	WIOSEC3	WIOSEC2	WIOSEC1
	Reset value																											0	0	0	0	0	0
0x184	**PWR_ WIOPRIVCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIOPRIV6	WIOPRIV5	WIOPRIV4	WIOPRIV3	WIOPRIV2	WIOPRIV1
	Reset value																											0	0	0	0	0	0
0x188 + 0x8*(x-1) (x=1 to 6) Last address: 0x1B0	**PWR_ WIOxCIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x18C + 0x8*(x-1) (x=1 to 6) Last address: 0x1B4	**PWR_ WIOxSEMCR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SEMCID[2:0]			Res	Res	Res	SEM_MUTEX
	Reset value																										0	0	0				0
0x1B8-0x1FC	Reserved	Reserved																															
0x200	**PWR_ CPU1D1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSTATE [2:0]			Res.	Res.	Res.	Res.	CSTATE [1:0]		Res.	HOLD_BOO
	Reset value																						0	0	0					0	0		0
0x204	**PWR_ CPU2D2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSTATE [2:0]			Res.	Res.	Res.	Res.	CSTATE [1:0]		WFBEN	HOLD_BOO
	Reset value																						0	0	0					0	0	0	0
0x20C-0x3F0	Reserved	Reserved																															
0x3F4	PWR_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	PWR_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x3FC	PWR_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

20 Reset and clock control (RCC)

The RCC manages the clock and reset generation for the complete circuit (see *Section 19: Power control (PWR)* for more details on operating modes).

In the whole document:

- The Cortex-A35 is also called CPU1 or MPU.
- The Cortex-M33 is also called CPU2 or MCU.

D1 domain holds CPU1 (supplied by V_{DDCPU}). D2 domain holds CPU2 and the majority of the peripherals (supplied by V_{DDCORE}).

20.1 RCC main features

- AMBA 5 AHB bus interface
- RIF (resource isolation framework) aware
- Reset:
  - Generation of local and system reset
  - Bidirectional pin reset (NRST) to reset external devices, or the circuit
  - Output pin reset (NRSTC1MS) to reset external mass-storage devices used by CPU1
- Clock generation:
  - Generation and distribution of clocks for the complete system
  - 5 separate PLLs (excluding external CPU1 and DDR ones):
    - Integer or fractional mode
    - Spread-spectrum function to reduce the amount of EMI peaks
    - Possibility to change the PLLs fractional ratios on the fly
  - Smart clock gating for reduction of power dissipation
  - 2 external oscillators:
    - HSE supporting a wide range of crystals (8 to 48 MHz)
    - LSE for the 32.768 kHz crystals
  - 3 internal oscillators:
    - HSI running around 64 MHz
    - MSI running around 16 MHz
    - LSI running around 32 kHz
  - Buffered clock outputs for external devices
  - HSI and MSI frequency monitors
- 2 independent interrupt interfaces (one dedicated to CPU1 and one dedicated to CPU2)
- 2 independent failure events (HSE and LSE)
- 2 independent events to wake up CPU1 and CPU2 (one dedicated for each)

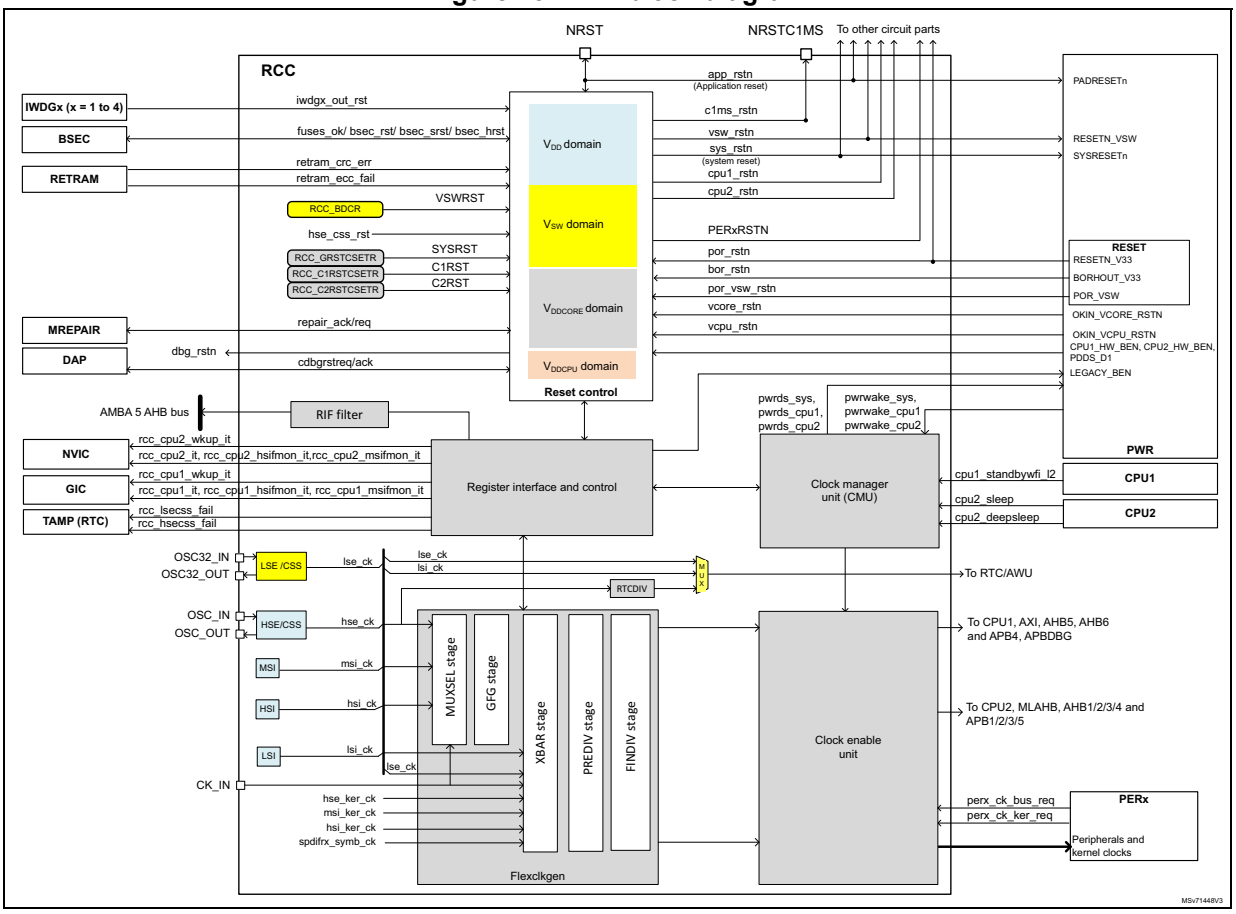
20.2 RCC block diagram

The figure below gives a simplified view of the RCC, and the key signals exchanged with the other peripherals. Four voltage domains are present in the RCC: V_{DD} domain, V_{SW} domain, V_{DDCPU} domain called D1 domain and V_{DDCORE} domain called D2 domain.

For the reset, analog comparators located into PWR provide all reset signals dependent of the power supplies: por_rstn, bor_rstn, vcore_rstn and por_vsw_rstn. One reset source comes from pins: NRST. The RCC also receives resets from various other sources (such as watchdogs, CPU2, or registers). All these resets are handled by the RCC in order to reset properly all system parts.

For the clock generation, the RCC handles the generation and the gating for all the clocks according to register settings and operating modes.

Figure 73. RCC block diagram



The RCC exchanges also some signals with the PWR in order to control the entry/exit of CPUx in CStop, D1 domain in DStandby mode, and system in Stop and Standby modes.

20.2.1 RCC pin overview

The table below lists the RCC signals connected to pins or balls.

Table 112. Pin overview

Pin name	Type	Description
NRST	Input/output	External (application) reset, can be used to provide reset to external devices
OSC32_IN	Input	32 kHz oscillator input
OSC32_OUT	Output	32 kHz oscillator output
OSC_IN	Input	System oscillator input
OSC_OUT	Output	System oscillator output
MCO1	Output	Output clock 1 for external devices
MCO2	Output	Output clock 2 for external devices
NRSTC1MS	Output	Output reset flag for external mass-storage devices used by CPU1
CK_IN	Input	External kernel clock input for digital audio interfaces: SPI/I2S, SAI, and MDF

20.3 RCC functional description - Reset

There are several sources able to generate a reset:

- supply monitors (V_{DD}, V_{DDCORE}, V_{DDCPU} or V_{SW}) lower than expected values
- independent watchdogs timeout
- D1 domain exiting DStandby
- device exiting Standby
- external signals driving the NRST pin
- software commands

The scope of the resets differs according to the source initiating the reset. The following categories are defined below: power-on/off, system, and local resets

20.3.1 Power-on/off resets

The PWR provides the following reset signals to the RCC:

- por_rstn signal, asserted when V_{DD} supply is lower than V_{POR} threshold
- bor_rstn signal, asserted when V_{DD} supply is lower than V_{BOR} threshold
- vcore_rstn signal, asserted when V_{DDCORE} supply is lower than V_{CORE} threshold
- vcpu_rstn signal, asserted when V_{DDCPU} supply is lower than V_{CPU} threshold
- por_vsw_rstn signal, asserted when V_{SW} supply is lower than the expected threshold

As shown in *Figure 74*, the signals provided by the PWR, indicating that V_{DDCORE} and V_{DDCPU} are valid (OKIN_VCORE_RSTN, OKIN_VCPU_RSTN), are used by the RCC in order to trigger the hardware system initialization sequence and the memory repair. When this initialization sequence is completed, vcore_rstn and vcpu_rstn signals are released, and CPU1/2 can start (see *Table 115* for details).

See *Section 19: Power control (PWR)* for details on the generation of power-on/off resets.

20.3.2 Application and system resets

An application reset (app_rstn) is generated from one of the following sources:

- reset from NRST pin
- reset from por_rstn signal (generally called power-on reset)
- reset from bor_rstn signal (generally called brownout reset)
- reset from the independent watchdogs (iwdg1_out_rst, iwdg2_out_rst, iwdg3_out_rst, iwdg4_out_rst) if the application allows them in RCC_IWDGC1CFGSETR, RCC_IWDGC2CFGSETR
- software reset from the RCC, when either CPU1 or CPU2 sets SYSRST in RCC_GRSTCSETR
- failure on HSE, when the clock security system feature is activated (hse_css_rst)
- RETRAM CRC error in wake-up event mode. After system Standby exit, when RET_CRCERR_RSTEN is set in RCC_STBYBOOTCR, and TDCID = 0x1. When TDCID = 0x2, it triggers an application reset whatever RET_CRCERR_RSTEN.
- RETRAM ECC failure during the CRC computation in wake-up event mode. After system Standby exit, when RET_CRCERR_RSTEN is set in RCC_STBYBOOTCR and TDCID = 0x1. When TDCID = 0x2, it triggers an application reset whatever RET_CRCERR_RSTEN.
- RETRAM ECC failure in wake-up event mode, after system Standby exit, and during system wake-up restoration phase, once CRC is done

A system reset (sys_rstn) is generated from one of the following sources:

- reset from app_rstn signal (application reset)
- reset from vcore_rstn signal
- reset from vcpu_rstn signal when the D1 domain does not exit Standby

Note: When the system is in Standby, V_{DDCORE} and V_{DDCPU} are switched off, but V_{DD} is still present. When the system exits Standby, vcore_rstn is activated, generating a sys_rstn reset. vcpu_rstn may also be activated depending on the selected system Standby exit strategy.

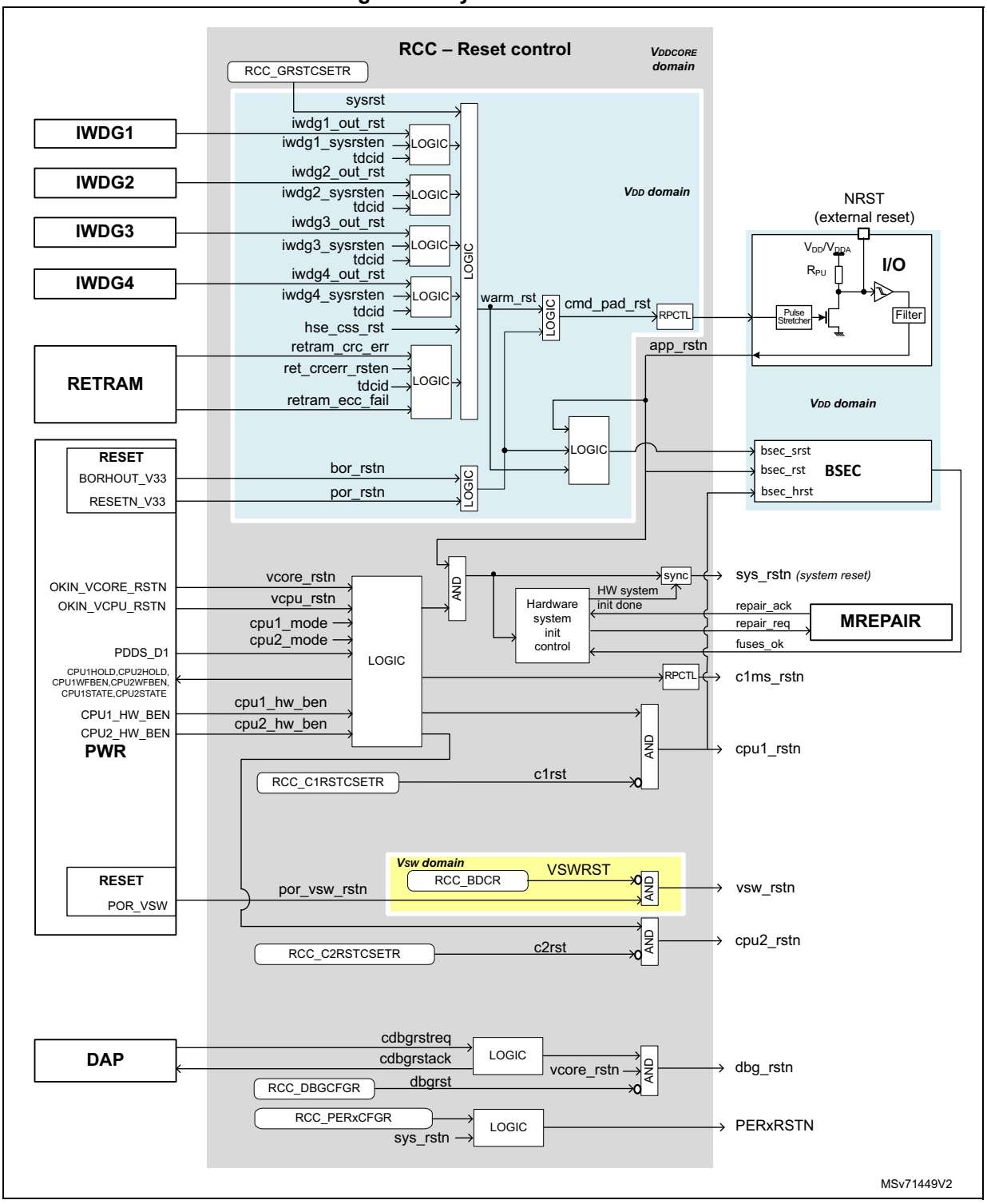
As shown in *Figure 74*, the NRST pin is activated by:

- por_rstn, bor_rstn, or hse_css_rst
- iwdg1_out_rst, iwdg2_out_rst, iwdg3_out_rst, iwdg4_out_rst according to the application program
- sysrst
- retram_crc_err, or retram_ecc_fail according to the application program
- assertion of NRST by an external source

A programmable reset pulse control (RPCTL) allows the application to define a minimum pulse width for the NRST assertion. When the RPCTL is disabled, the reset pin embeds a pulse stretcher which guarantees a minimum reset pulse duration of 20 microseconds for each internal reset source. app_rstn and sys_rstn are also impacted by this function (see *Section 20.3.10* for details on RPCTL).

Note: It is not recommended to let the NRST pin unconnected. When it is not used, connect this pin to ground via a capacitor of 10 to 100 nF.

Figure 74. System reset circuit



20.3.3 Local resets

The local resets do not directly affect the whole system, but only one CPUx subsystem (cores and its associated peripherals, like GIC or NVIC).

Here is the most important local reset:

When the D1 domain exits DStandby, and the system was not in Standby, the CPU1 is reset by a reset named cpu1_rstn, and the external mass-storage devices used by CPU1 can be reset by the c1ms_rstn signal (NRSTC1MS pin). Refer to *NRSTC1MS I/O* for details.

20.3.4 Resets generated by CPUs

All CPUs can generate system and local resets. The RCC offers the following possibilities:

- CPUx can generate a system reset by means of SYSRST in RCC_GRSTCSETR.
- C2RST in RCC_C2RSTCSETR can reset CPU2 and its related watchdogs (IWDG3/4).
- C1RST in RCC_C1RSTCSETR can reset CPU1 and its related watchdogs (IWDG1/2).

The CPU1 is reset whenever the D1 domain exits DStandby.

When the CPU2 uses C1RST to reset the CPU1, the following procedure must be used:

1. Clear RCC_CPUBOOTCR.BOOT_CPU1.
2. Activate RCC_C1RSTCSETR.C1RST.
3. Poll RCC_C1RSTCSETR.C1RST until it is 0.
4. Set RCC_CPUBOOTCR.BOOT_CPU1.

20.3.5 Watchdog resets

The following watchdogs are provided to the application:

- IWDG1 (secure) and IWDG2 (non-secure) dedicated to the CPU1
- IWDG3 (secure) and IWDG4 (non-secure) dedicated to the CPU2

All these watchdogs can generate a system reset when a timeout occurs, according to the application selection.

The system reset operation is controlled by (see *Table 113* and *Table 114* for details):

- IWDG1_SYSRSTEN, IWDG2_SYSRSTEN in RCC_IWDGC1CFGSETR
- IWDG3_SYSRSTEN, IWDG4_SYSRSTEN in RCC_IWDGC2CFGSETR
- TCID selected trusted domain master agent

Warning: The application must set these IWDGx_SYSRSTEN bits prior enabling any watchdog.

These registers are located in V_{DD} domain and are reset by app_rstn.

Table 113. IWDG timeout event combinations when TDCID = CPU1

IWDG1_ SYSRSTEN(1)	IWDG2_ SYSRSTEN	IWDG3_ SYSRSTEN	IWDG4_ SYSRSTEN	Timeout event description
-	0	0	0	System reset from IWDG1
			1	System reset from IWDG1 and IWDG4
	1		0	System reset from IWDG1 and IWDG2
			1	System reset from IWDG1, IWDG2, and IWDG4
	0	1	0	System reset from IWDG1 and IWDG3
			1	System reset from IWDG1, IWDG3, and IWDG4
	1		0	System reset from IWDG1, IWDG2, and IWDG3
			1	System reset from all watchdogs

1. "-" means don't care.

Table 114. IWDG timeout event combinations when TDCID = CPU2

IWDG1_ SYSRSTEN	IWDG2_ SYSRSTEN	IWDG3_ SYSRSTEN(1)	IWDG4_ SYSRSTEN	Timeout event description
0	0	-	0	System reset from IWDG3
			1	System reset from IWDG3 and IWDG4
	1		0	System reset from IWDG2 and IWDG3
			1	System reset from IWDG2, IWDG3, and IWDG4
1	0		0	System reset from IWDG1 and IWDG3
			1	System reset from IWDG1, IWDG3, and IWDG4
	1		0	System reset from IWDG1, IWDG2, and IWDG3
			1	System reset from all watchdogs

1. "-" means don't care.

In addition, any IWDG timeout event is sent to the main EXTI to be routed to all CPUs.

20.3.6 Backup domain reset

A backup domain reset (vsw_rstn) is generated when one of the following events occurs:

- software reset, triggered by setting VSWRST in RCC_BDCR

  All RTC registers and RCC_BDCR are set to their reset values. The BKPSRAM is not affected.
- when V_{SW} voltage is outside the operating range (por_vsw_rstn)

  All RTC registers and RCC_BDCR are set to their reset values. The BKPSRAM is no longer valid.

The por_rstn reset source for the V_{DD} domain is gated by the por_vsw_rstn reset in order to secure the internal power sequence. See *Section 19: Power control (PWR)* for additional information.

20.3.7 CoreSight debug reset

The CoreSight debug components including the CPU1 debug part can be reset in three different ways:

- over the DAP by setting to 1 CDBGRSTREQ in control/status register of the debug port
  Setting CDBGRSTREQ = 1 asserts the debug reset request signal CDBGRSTREQ connected to the RCC. The RCC activates dbg_rstn. An handshake signal cdbgrstack is provided to the DAP request. dbg_rstn is active as long as the CDBGRSTREQ = 1.
- by the application by setting to 1 DBGRST in RCC_DBGCFGR
  When DBGRST = 1, the dbg_rstn reset is activated. dbg_rstn reset is released when DBGRST = 0.
- when V_{DDCORE} power-on reset occurs (vcore_rstn)
  This reset is activated after a power-on reset, or when the product exits Standby.

20.3.8 Option-byte loading

The boot and security register interface module (BSEC) controls an OTP (one-time programmable) array of fuse words that hold the option-byte configuration for the product.

This configuration must be set every time an app_rstn or a NRST reset is activated. The system reset is kept active until this configuration has been properly loaded (via the fuse_ok signal received from the BSEC).

The BSEC handles the following reset sources (see *Section 6: Boot and security control (BSEC)* for details):

- a main reset (bsec_rst) active when an app_rstn or a NRST reset is triggered
- a scratch reset (bsec_srst) active when a por_rstn, bor_rstn or a NRST reset is triggered
- a hot reset (bsec_hrst) active when a CPU1 reset is triggered

Note: When a warm reset occurs, any NRST reset is filtered, and the bsec_srst reset is not performed.

20.3.9 Peripheral reset

The application can reset individually any peripheral whenever it is requested. This can be done by CPUx via RCC_PERxCFGR (with PERx = peripheral name).

Note: Access rights to these registers are inherited from the related peripheral RIF configuration (see Section 20.7.1 for details).

When a peripheral needs to be reset, it is recommended to first enable its peripheral clock by setting the corresponding peripheral enable bit to 1 (see *Section 20.4.11* for details).

For peripherals having a reset process which is executed very fast, it is up to the application to assert and release the peripheral reset (in RCC_PERxCFGR, with PERx = peripheral name).

For some peripherals, the reset process may take several clock cycles. For that purpose, the application can read back the reset bit in order to check if the reset process is ongoing or completed.

Note: DCMIPP, ETH1, ETH2, HPDMA1, HPDMA2, HPDMA3, LTDC, SDMMC1, SDMMC2, SDMMC3, USBH and OTG peripherals are following this above reset behavior: when the

application releases the peripheral reset, it must read back the related PERxRST in RCC_PERxCFGR (with PERx = peripheral name) to be sure that the reset has been properly released. This applies when the CFG_ENABLE bit is set in SYSCFG_SAFERSTCR.

20.3.10 Reset pulse control (RPCTL)

The RPCTL allows the application to control the minimum activation time of NRST and NRSTC1MS pins. This feature is particularly helpful because some external devices may request a specific duration for the activation of these pins. In addition, the internal reset pulse like IWDGx may be too short for external devices.

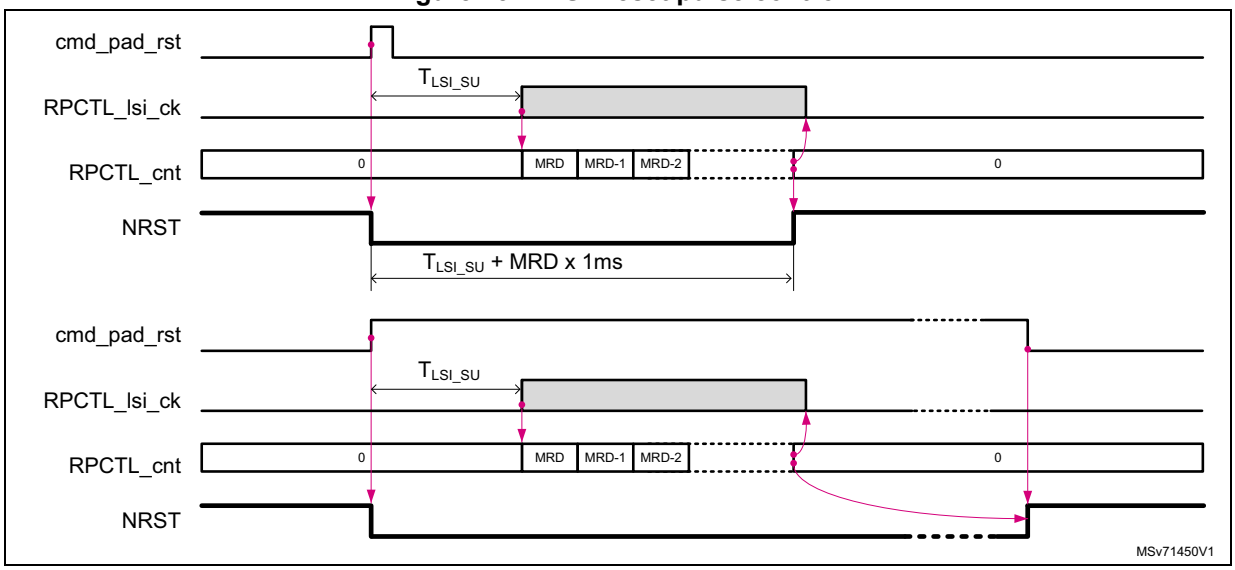
NRST pin

The RPCTL is located into the V_{DD} domain, and is only reset after a power-on reset. It is controlled by MRD[4:0] in RCC_RDCR as follows:

- If MRD = 0, the RPCTL is bypassed, and the cmd_pad_rst signal directly drives the reset pin. The minimum activation time is given by the pulse stretcher embedded into the reset pin (typically 20 microseconds).
- If MRD = 1, when the reset is activated (cmd_pad_rst rising edge), NRST is immediately driven low at least for a duration equal to the value given by MRD[4:0]. The reset duration is not affected by RPCTL when the cmd_pad_rst signal is active for a duration longer than the value programmed in MRD[4:0]. The MRD[4:0] is used to adjust the minimum activation time between 1 to 31 ms.

Note: The RPCTL uses the LSI clock in order to control the reset width. When cmd_pad_rst goes high, the RPCTL requests the LSI clock in order to control the reset duration. If the LSI was not enabled by another function, it may take some microseconds before having the LSI ready (T_{LSI_SU}).

Figure 75. NRST reset pulse control

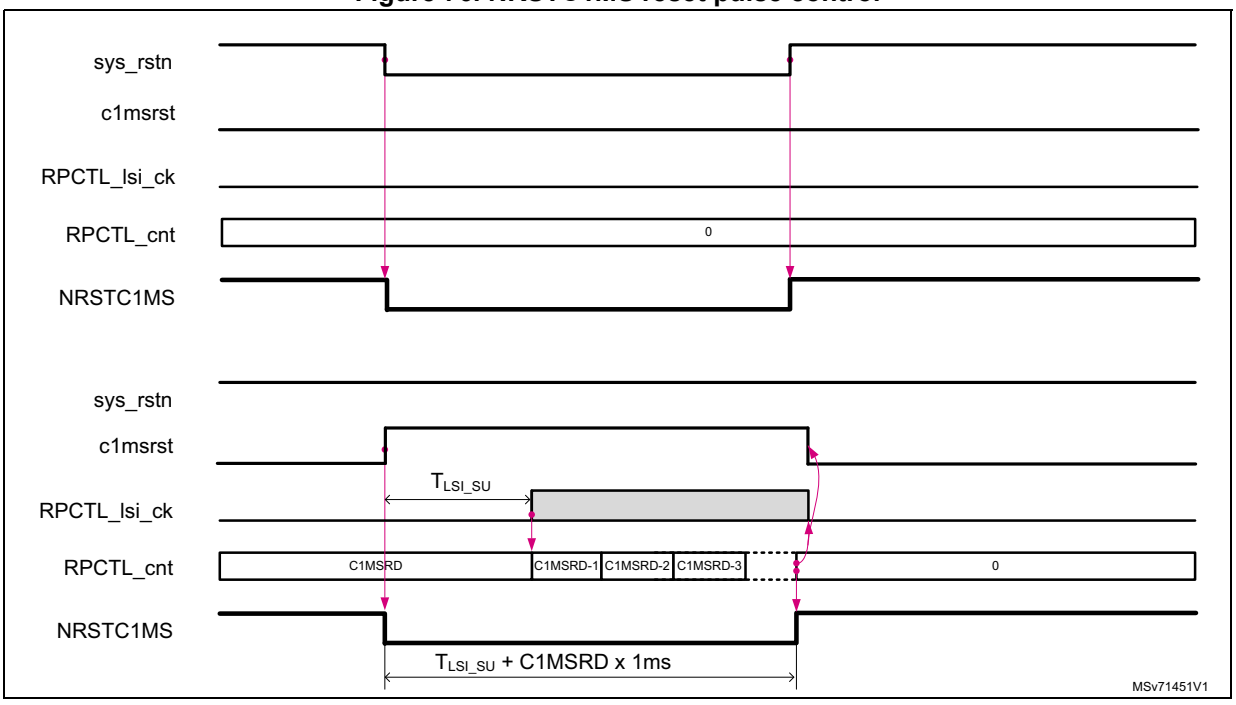


NRSTC1MS I/O

The RPCTL is the same as the NRST except the following:

- It is located into the V_{DDCORE} domain and the NRSTC1MS pin is open drain.
- It is reset after a system reset.
- It is controlled by C1MSRST and C1MSRD[4:0] in RCC_C1MSRDCR.

Figure 76. NRSTC1MS reset pulse control



20.3.11 Reset coverage summary

The table below gives a detailed view of reset sources and reset domains. Each column represents a reset source.

Note: It is assumed that when V_{DD} is not valid, V_{DDCORE} and V_{DDCPU} are not valid.

Table 115. Reset coverage summary

Reset items	Reset lines[1]									
	por_rstn[2]	vcore_rstn	sys_rstn	app_rstn	vcpu_rstn	cpu1_rstn	cpu2_rstn	dbg_rstn	perx_rstn	vsw_rstn
V_{DD} domain	x	-	-	-	-	-	-	-	-	-
DDR ITF	x	x	x	x	-	-	-	-	-	-
CPU1 (D1 domain)	x	-	x	x	x	x	-	-	-	-
CPU2 (D2 domain)	x	x	x	x	-	-	x	-	-	-
AXI/AHB interconnections (D2 domain)	x	x	x	x	-	-	-	-	-	-
CoreSight debug components: reset all the debug parts including the ones located in CPU1 except SWJ-DP function. The SWJ-DP is reset by NTRST or an app_rstn reset.	x	x	x	x	-	-	-	x	-	-

Table 115. Reset coverage summary (continued)

Reset items		Reset lines[1]									
		por_rstn[2]	vcore_rstn	sys_rstn	app_rstn	vcpu_rstn	cpu1_rstn	cpu2_rstn	dbg_rstn	perx_rstn	vsw_rstn
IWDG1		x	-	-	x	-	x	-	-	-	-
IWDG2		x	-	-	x	-	x	-	-	x	-
IWDG3		x	-	-	x	-	-	x	-	-	-
IWDG4		x	-	-	x	-	-	x	-	x	-
Hardware system init. It includes the memory repair.		x	x	-	x	x	-	-	-	-	-
RCC	RCC_LEGBOOTCR, RCC_STBYBOOTCR, RCC_LSICR	x	-	-	x	-	-	-	-	-	-
	RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, RCC_C2HWRSTSCLRR	x	-	-	-	-	-	-	-	-	-
	RCC_IWDGC1FZSETR	x	x	x	x	x	x	-	-	-	-
	RCC_IWDGC2FZSETR	x	x	x	x	-	-	x	-	-	-
	RCC_RDCR	x	-	-	-	-	-	-	-	-	-
	RCC_BDCR	-	-	-	-	-	-	-	-	-	x
	Other RCC registers	x	x	x	x	-	-	-	-	-	-
PWR	PWR_CR2, PWR_CR9, PWR_CR10	-	-	-	-	-	-	-	-	-	x
	PWR_CR5, PWR_CR6, PWR_WKUPCRx	x	-	-	x	-	-	-	-	-	-
	PWR_D1CR	x	-	-	-	-	-	-	-	-	-
	PWR_CPU1CR, PWR_CPU2CR, PWR_D2CR Individual bits of these registers do not have the same reset condition (see *Section 19: Power control (PWR)* for details).	x	x	x	x	-	-	-	-	-	-
	Other registers	x	x	x	x	-	-	-	-	-	-
RTC		-	-	-	-	-	-	-	-	-	x
BKPSRAM and RETRAM After a reset of the V_{SW} domain, BKPSRAM and RETRAM backup regulators are disabled. This is controlled BKPRBSEN in PWR_CR9, RETRBSEN in PWR_CR10 respectively. If the vsw_rstn reset is due to a too low V_{SW} voltage, the content of BKPSRAM and RETRAM is lost.		-	-	-	-	-	-	-	-	-	x
Central RIF components		x	x	x	x	-	-	-	-	-	-
Other peripherals		x	x	x	x	-	-	-	-	x	-

1. The reset lines that correspond reset item are highlighted in gray.
2. When por_rstn is asserted, all resets are asserted as well except perx_rstn and vsw_rstn.

Reset of RCC registers

The table below lists registers having bits into a voltage domain different from V_{DDCORE}. As a consequence some of these registers are not always reset by sys_rstn. The RCC registers not shown in the table below are reset by sys_rstn.

Table 116. RCC registers with specific reset and specific voltages

RCC register name	Voltage domain	Reset condition	Comment
RCC_LEGBOOTCR, RCC_STBYBOOTCR, RCC_LSICR, RCC_IWDGC1CFGSETR, RCC_IWDGC1CFGCLRR, RCC_IWDGC2CFGSETR, RCC_IWDGC2CFGCLRR	V_{DD}	app_rstn	Must be reset after a system reset, but not after Standby reset
RCC_OCENSETR, RCC_OCENCLRR, RCC_OCRDYR, RCC_HSICFGR, RCC_MSICFGR		sys_rstn	-
RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, RCC_C2HWRSTSCLRR, RCC_RDCR		por_rstn	Only reset when a V_{DD} is removed
RCC_BDCR	V_{SW}	vsw_rstn	VSWRST not reset by por_vsw_rstn

20.3.12 Reset source identification

The RCC offers the following registers in order to identify the root cause of reset:

- *RCC hardware reset status clear register (RCC_HWRSTSCLRR)*
- *RCC CPU1 hardware reset status clear register (RCC_C1HWRSTSCLRR)*
- *RCC CPU2 hardware reset status clear register (RCC_C2HWRSTSCLRR)*
- *RCC CPU1 boot reset status set register (RCC_C1BOOTRSTSSETR)*
- *RCC CPU1 boot reset status clear register (RCC_C1BOOTRSTSCLRR)*
- *RCC CPU2 boot reset status set register (RCC_C2BOOTRSTSSETR)*
- *RCC CPU2 boot reset status clear register (RCC_C2BOOTRSTSCLRR)*

The first booting CPUx can access RCC_HWRSTSCLRR to identify the reset root cause. It also relies on PWR_CPUxCR content (see *Section 19: Power control (PWR)* for details) to set which kind of boot sequence must be performed after any kind of reset. RCC_C1HWRSTSCLRR and RCC_C2HWRSTSCLRR are respectively dedicated to CPU1 and CPU2 to identify their own root cause of reset.

When CPU1 first booting, the boot ROM uses RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, and PWR_CPU1CR to set the boot reset reason via RCC_C1BOOTRSTSSETR after every system reset, power-on reset, or exit from Standby. The CPU1 can then check the boot reason and clear the flags via RCC_C1BOOTRSTSCLRR.

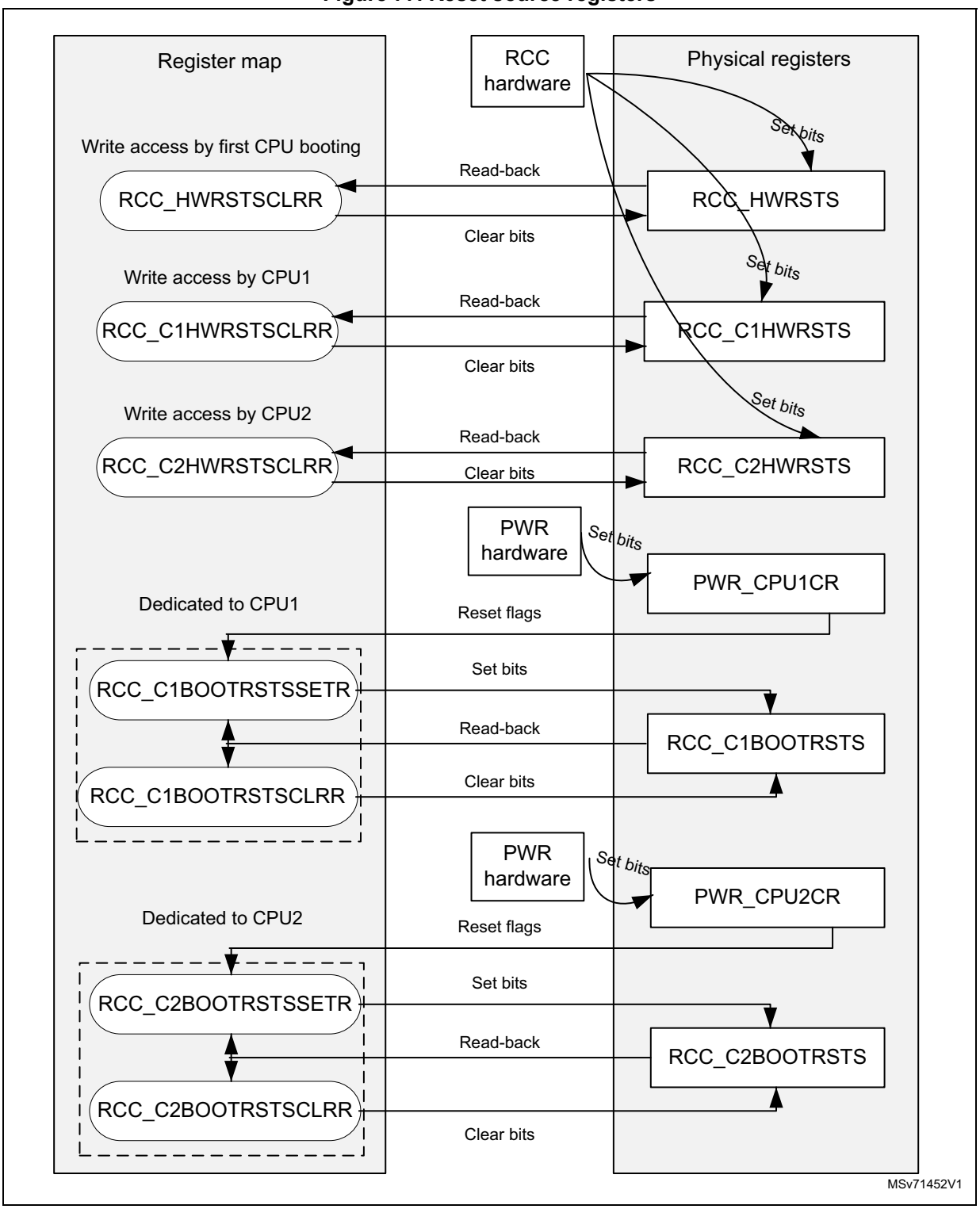
When CPU2 first booting, the CPU2 boot code uses also RCC_HWRSTSCLRR, RCC_C2HWRSTSCLRR, and PWR_CPU2CR to set the boot reset reason via RCC_C2BOOTRSTSSETR.

Note: STBYC1RSTF, STBYC2RSTF, D2STBYRSTF and D1STBYRSTF flags in RCC_C1BOOTRSTSSETR and RCC_C2BOOTRSTSSETR are copied from SBF, SBF_D2 and SBF_D1 in PWR_CPU1CR and PWR_CPU2CR.

There is not any hardware protection on bit clear inside RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, and RCC_C2HWRSTSCLRR. Once the boot reset reason has

been set, the application (CPU1 or/and CPU2) must clear RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, and RCC_C2HWRSTSCLRR.

Figure 77. Reset source registers



The table below shows the status bits in RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, and RCC_C2HWRSTSCLRR according to the reset source.

Table 117. RCC HWRSTSCLRR, RCC_C1HWRSTSCLRR, RCC_C2HWRSTSCLRR flags(1)

#	Reset source	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	C2RSTF	C1RSTF	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCPURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
1	Power-on reset (por_rstn)	-	-	-	1	1	-	-	-	-	-	-	1	1	-	1	-	1
2	Brownout reset (bor_rstn)	-	-	-	1	1	-	-	-	-	-	-	-	-	-	1	1	-
3	Pin reset from NRST	-	-	-	1	1	-	-	-	-	-	-	-	-	-	1	-	-
4	Clock failure on HSE	-	-	-	1	1	-	-	-	-	-	-	-	-	1	1	-	-
5	System reset (SYSRST) by CPU1	-	-	-	1	1	-	-	-	-	-	1	-	-	-	1	-	-
6	System reset (SYSRST) by CPU2	-	-	-	1	1	-	-	-	-	1	-	-	-	-	1	-	-
7	System reset from V_{CPU} monitor	-	-	-	1	1	-	-	-	-	-	-	1	-	-	-	-	-
8	CPU1 reset by CPU2 (C1RST)(2)	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
9	CPU2 reset by CPU1 (C2RST)(3)	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
10	System exits Standby in cold boot CPU2 mode without error	-	-	-	1	-	-	-	-	-	-	-	-	1	-	-	-	-
11	System exits Standby in other modes(4) without error	-	-	-	1	1	-	-	-	-	-	-	1	1	-	-	-	-
12	System exits Standby with RETRAM CRC error	-	-	1	1	1	-	-	-	-	-	-	1	1	-	1	-	-
13	System exits Standby with RETRAM ECC failure during CRC	-	1	-	1	1	-	-	-	-	-	-	1	1	-	1	-	-
14	System exits Standby with RETRAM ECC failure during system restoration	1	-	-	1	1	-	-	-	-	-	-	1	1	-	1	-	-
15	D1 domain exits DStandby	-	-	-	-	1	-	-	-	-	-	-	1	-	-	-	-	-
16	IWDG1 system reset (iwdg1_out_rst)	-	-	-	1	1	-	-	-	1	-	-	-	-	-	1	-	-
17	IWDG2 system reset (iwdg2_out_rst)	-	-	-	1	1	-	-	1	-	-	-	-	-	-	1	-	-
18	IWDG3 system reset (iwdg3_out_rst)	-	-	-	1	1	-	1	-	-	-	-	-	-	-	1	-	-
19	IWDG4 system reset (iwdg4_out_rst)	-	-	-	1	1	1	-	-	-	-	-	-	-	-	1	-	-

1. The 1 status is highlighted in grey.
2. This bit is set when C1RST is activated in RCC_C1RSTCSETR.
3. This bit is set when C2RST is activated in RCC_C2RSTCSETR.
4. VCPURSTF is present only when D1 is not in kept in DStandby.

20.3.13 Power-on and wake-up sequences

For detailed diagrams refer to *Section 19: Power control (PWR)*.

The time interval between the event which exits the product from a low-power mode and the moment where the CPUs are able to execute code, depends on the state of the system and its configuration. *Figure 78* shows the most usual examples.

Power-on wake-up sequence

The power-on wake-up sequence of *Figure 78* gives the most significant phases of the power-on sequence. It is the longest sequence as the circuit was not powered.

Note: This sequence remains unchanged if V_{BAT} was present or not.

Boot sequence from pin reset (NRST)

The pin reset gives the most important steps of the boot sequence.

Note: In this case, V_{DD}/V_{DDA} have not been removed.

- The setting time is faster as the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI was not enabled when the NRST occurred. Otherwise this restart delay phase is skipped.
- The option byte reloading is performed.

Boot sequence from system Standby

For a wake-up from system Standby, as the V_{DD} is not removed, the settling time is also fast as the reference voltage is already stable. As V_{DDCORE} and V_{DDCPU} were not there, the restart delay for the HSI cannot be skipped.

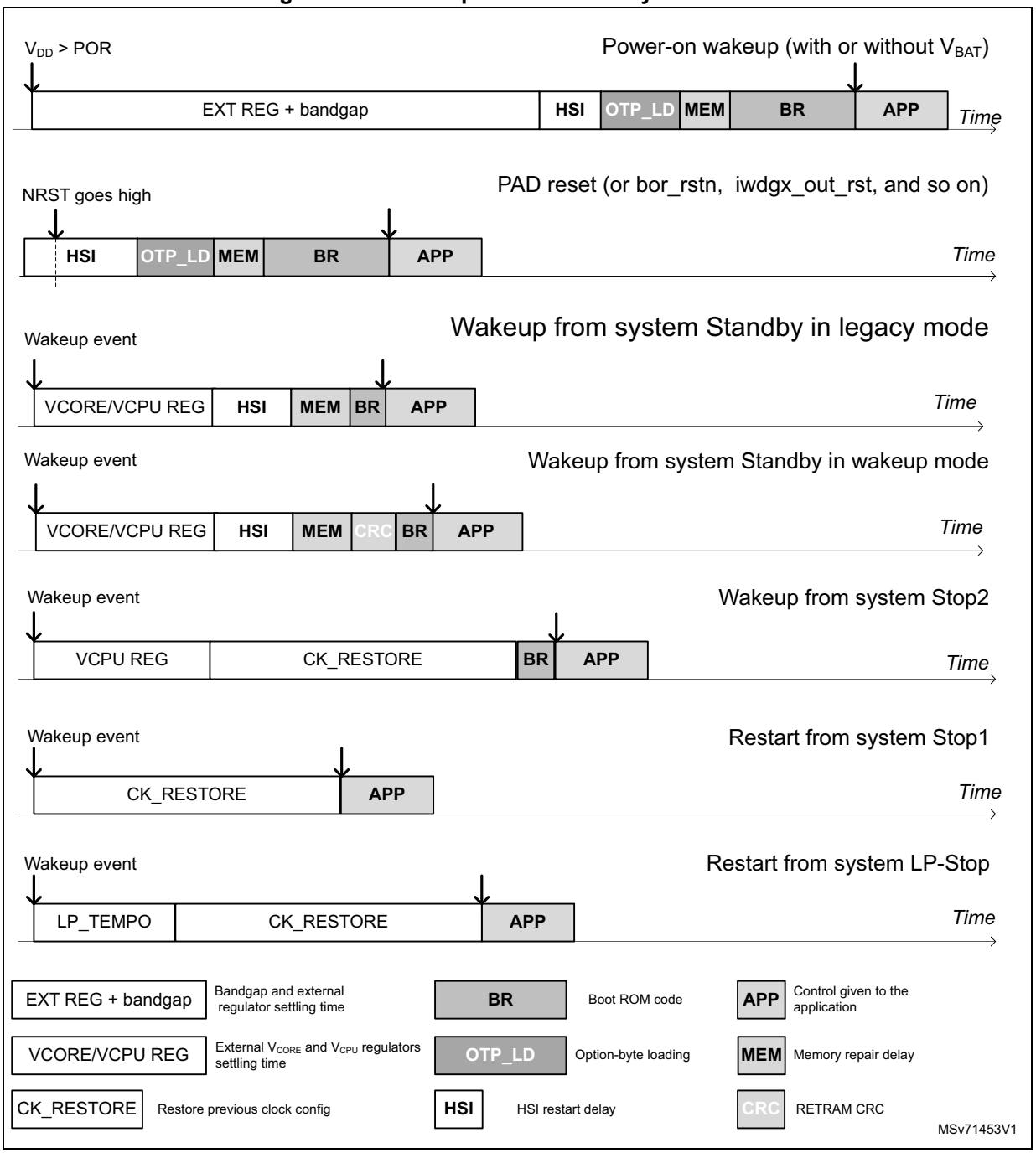
In the cold boot CPU2 and wake-up event modes, when the system exits Standby, the CRC is run on the RETRAM content, whatever the wake-up source. This fastens the CPU2 firmware execution in case of CPU2 wake-up event.

Restart sequence from system Stop

For a restart from system Stop1, both V_{DDCORE} and V_{DDCPU} have not been removed, so the settling time is mainly due to the restart delay of the clock. The system restarts according to the clock restore sequence (see *Clock restore sequence* for details).

For a restart from system Stop2, in case of CPU1 wake-up event, the boot ROM code execution time is added.

Figure 78. Boot sequences versus system states



CPUx state after application reset

After an application reset (from IWDG, POR, BOR, or pin), the CPU1 always starts the boot ROM code execution. The boot ROM code selects which CPU has to boot first (CPU1 or CPU2) while the other CPU is maintained on HOLD_BOOT state. The CPUx selected to boot first is called cold boot CPU.

This HOLD_BOOT state is controlled via BOOT_CPU1 and BOOT_CPU2 in RCC_CPUBOOTCR (see *HOLD_BOOT CPUx state* for more details).

CPUx state after Standby exit

After any system Standby exit (vcore_rstn and vcpu_rstn are low), the product can select which CPUx (CPU1 or CPU2) boots first.

The selection of the first CPUx to boot depends on the following criteria:

- LEGACY_BEN, COLD_CPU2, and CPU_BEN_SEL in RCC_LEGBOOTCR and RCC_STBYBOOTCR

  They are reset by hardware after an application reset (from IWDG, POR, BOR, or pin), but not when the product exits Standby. These bits are located in the V_{DD} domain in order to be reset after a power-on reset.
- cpu1_hw_ben and cpu2_hw_ben from the PWR (see *Section : PWR entering Standby mode*)

There are different behaviors regarding this selection process.

- Legacy mode for first CPUx selection boot

  This mode is active when LEGACY_BEN is set to 1. In this mode, only CPU1 is allowed to boot, thus to execute the boot ROM code, while CPU2 is always maintained on HOLD_BOOT by the hardware.

  In order to allow CPU2 to boot, CPU1 must only set BOOT_CPU2 in RCC_CPUBOOTCR.
- Cold boot CPU2 mode for first CPUx selection boot

  This mode is active when LEGACY_BEN is cleared to 0 and COLD_CPU2 is set to 1. In this mode, only CPU2 can boot, while CPU1 is always maintained on HOLD_BOOT by the hardware. In case of RETRAM CRC or ECC failure, an application reset is triggered.

  In order to allow CPU1 to boot, CPU2 must set BOOT_CPU1 located in RCC_CPUBOOTCR and CPU1_HW_BEN must be active.
- Wake-up event mode for first CPUx selection boot

  This mode is active when LEGACY_BEN and COLD_CPU2 are cleared to 0. In this mode, the selection of the first CPUx to boot is based on CPU1_HW_BEN and CPU2_HW_BEN which are set according to the Standby exit wake-up events definition done in the PWR (see *Section 19: Power control (PWR)* for details about CPU1_HW_BEN and CPU2_HW_BEN).

  Example: if a CPU2 wake-up event occurs, the CPU2 boots on the authenticated RETRAM (based on hardware CRC engine), and restores the product security configuration, while CPU1 is kept on HOLD_BOOT state (see *HOLD_BOOT CPUx state* for details).

Table 118. First CPUx boot selection

LEGACY_BEN	COLD_CPU2	CPU_BEN_SEL	CPU1_HW_BEN	CPU2_HW_BEN	First CPUx boot selection
1	0	NA	0	0	CPU1
			1	0	
			0	1	
			1	1	

Table 118. First CPUx boot selection (continued)

LEGACY_BEN	COLD_CPU2	CPU_BEN_SEL	CPU1_HW_BEN	CPU2_HW_BEN	First CPUx boot selection
0	0	NA	1	0	CPU1
			0	1	CPU2
		0	0	0	CPU1
		1			
		0	1	1	CPU1
		1			CPU2
	1	NA	NA	NA	CPU2

HOLD_BOOT CPUx state

The RCC is used to put either CPU1 or CPU2 on HOLD_BOOT every time the CPUx is reset by a CPU reset (cpu1_rstn or cpu2_rstn).

This function is controlled by BOOT_CPU1 and BOOT_CPU2 in *Section 20.8.27*.

For example, after a CPU2 reset (cpu2_rstn), if BOOT_CPU2 = 0, the CPU2 is kept on HOLD_BOOT state. Else CPU2 is allowed to boot.

Having one CPUx on HOLD_BOOT does not prevent the system to go to Stop or Standby mode. For example, when CPU2 is on HOLD_BOOT, the system is in Stop if CPU1 goes to CStop, then if CPU1 allows the Standby mode (if PDDS bits are set to 1), then the system goes to Standby. The same behavior applies for CPU2 when CPU1 is on HOLD_BOOT.

Figure 79 shows an example of CPU2 reset (cpu2_rstn).

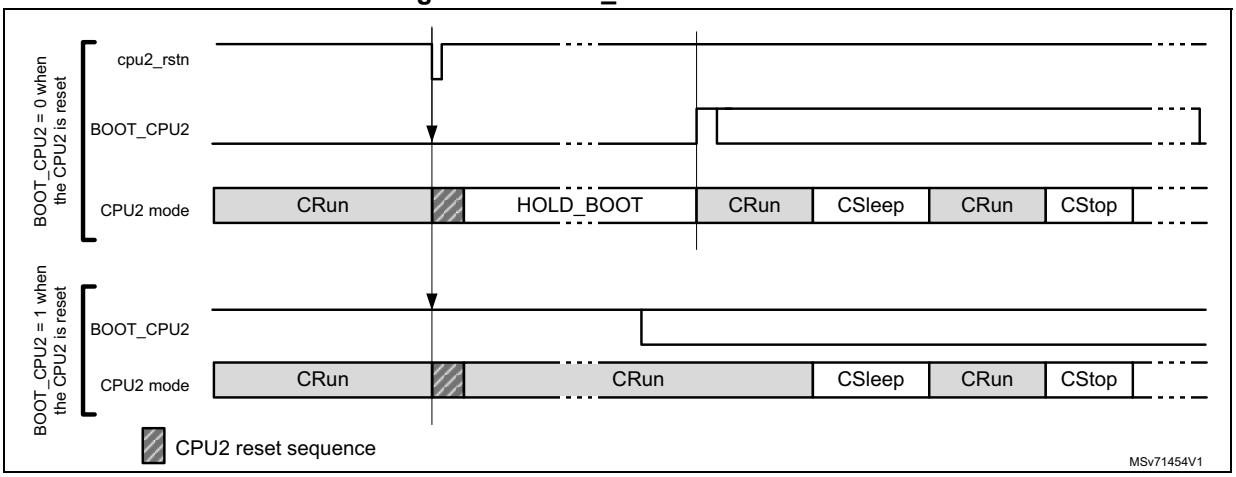
If BOOT_CPU2 = 0, the CPU2 stays in HOLD_BOOT mode until CPU1 sets BOOT_CPU2 to 1.

If BOOT_CPU2 = 1, the CPU2 switches immediately to CRun mode as soon as the CPU2 hardware reset sequence is completed.

In that case, the HOLD_BOOT state is equivalent to CStop. This means that when the CPUx is in HOLD_BOOT, all peripherals allocated by it, are no longer clocked, except the ones able to request a kernel clock, or working with lse_ck or lsi_ck.

Note: Once CPU2 is in CRun, the BOOT_CPU2 bit can be cleared again to 0 without disturbing the CPU2 activity. In that way, the CPU2 is again set in HOLD_BOOT state the next time it is reset via cpu2_rstn. It also applies to CPU1.

Figure 79. HOLD_BOOT CPUx reset



HOLD_BOOT CPUx state after D1 Standby exit

In this case, the CPU1 HOLD_BOOT state is controlled by BOOT_CPU1 in RCC_CPUBOOTCR.

If BOOT_CPU1 = 1, the CPU1 reset is released to allow its boot sequence.

HOLD_BOOT CPUx state after system Standby exit

In this case, HOLD_BOOT state depends on the selection mode of the first CPUx to boot.

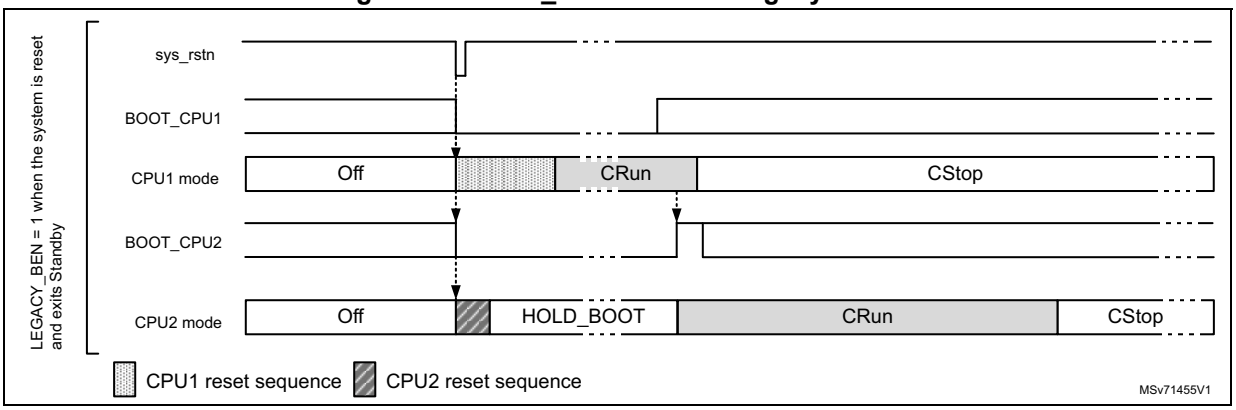
Table 119. HOLD_BOOT state control after system Standby exit

LEGACY_BEN	CPU1 HOLD_BOOT control	CPU2 HOLD_BOOT control
0	CPU1_HW_BEN(1), RCC_STBYBOOTCR.COLD_CPU2, RCC_STBYBOOTCR.CPU_BEN_SEL and RCC_CPUBOOTCR.BOOT_CPU1	CPU2_HW_BEN(1), RCC_STBYBOOTCR.COLD_CPU2, RCC_STBYBOOTCR.CPU_BEN_SEL and RCC_CPUBOOTCR.BOOT_CPU2
1	None	RCC_CPUBOOTCR.BOOT_CPU2(2)

1. CPU1_HW_BEN and CPU2_HW_BEN mirror the wake-up events defined in the PWR to exit system Standby. They can be read in RCC_STBYBOOTCR.
2. The reset value of BOOT_CPU2 applies in this case (see *Legacy mode for first CPUx selection boot*).

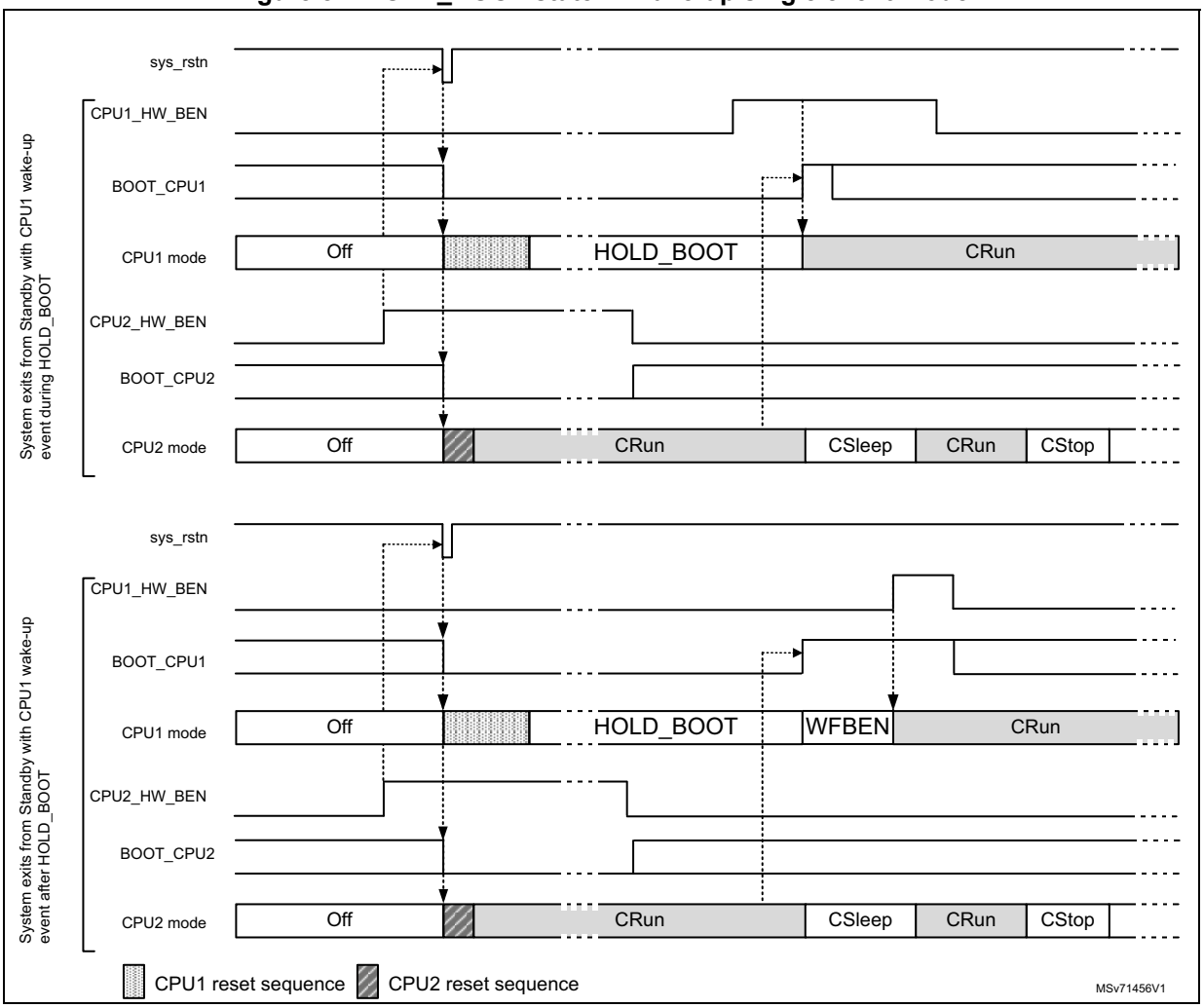
In legacy mode, the HOLD_BOOT state is controlled in the same way as the one described in *HOLD_BOOT CPUx state*.

Figure 80. HOLD_BOOT state in legacy mode



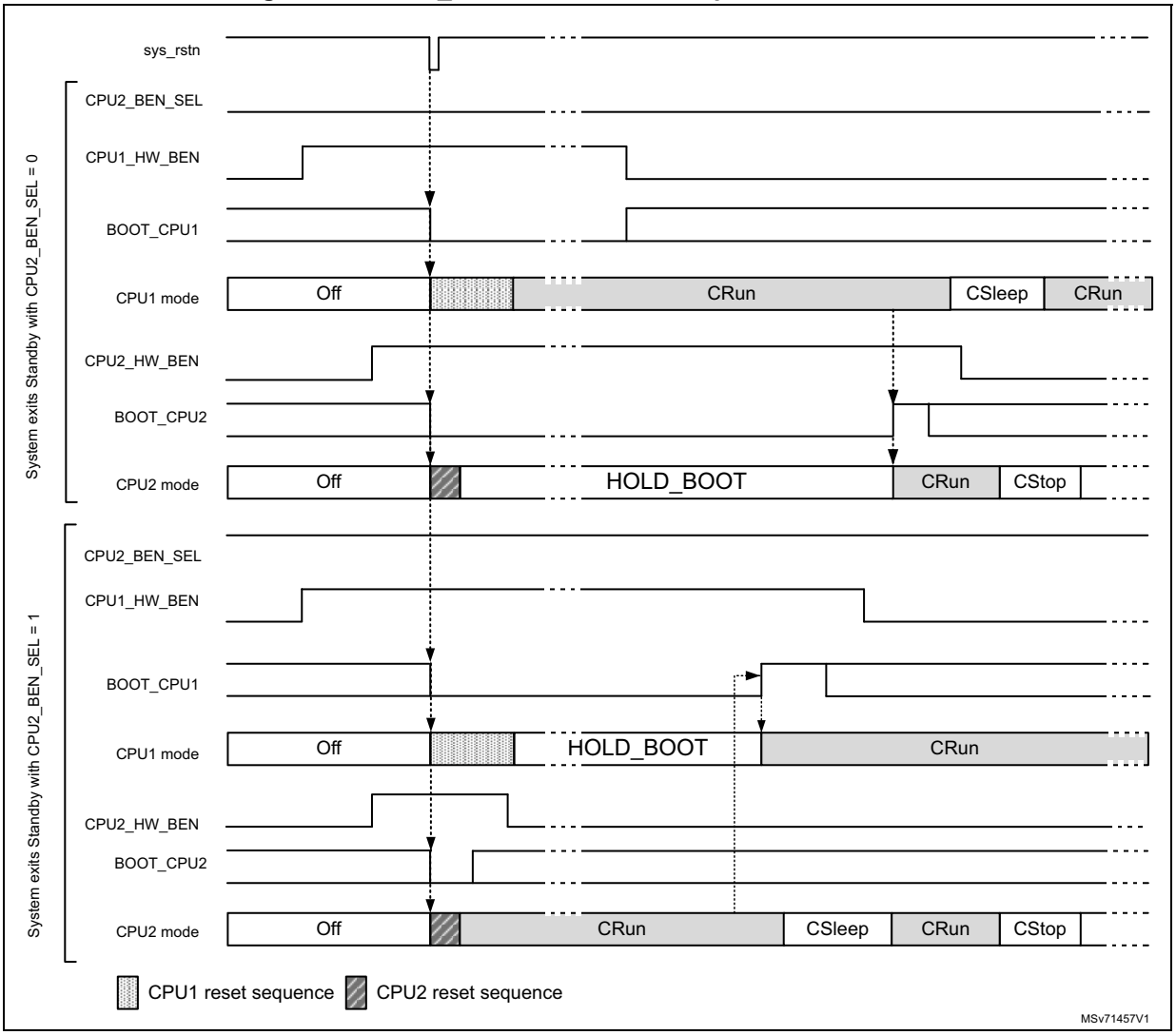
In wake-up event mode, the HOLD_BOOT state is dependent on CPU1_HW_BEN and CPU2_HW_BEN.

Figure 81. HOLD_BOOT state in wake-up single event mode



In this example, the CPU2 is selected first, and it can release the CPU1 HOLD_BOOT state by setting the BOOT_CPU1 bit.

Figure 82. HOLD_BOOT state in wake-up dual event mode



Note: Once a CPU is in CRun, BOOT_CPU1 and BOOT_CPU2 bits can be modified without disturbing the CPU activity.

HOLD_BOOT and WFBEN status for both CPUs are accessible in PWR_CPU1D1SR and PWR_CPU2D2SR (see *Section 19: Power control (PWR)* for details).

20.4 RCC functional description - Clocks

The RCC provides a high degree of flexibility to the application in the choice of the clock generators:

- clock from HSI (high-speed internal) oscillator (~ 64 MHz)
- clock from HSE (high-speed external) oscillator (8 to 48 MHz)
- clock from LSE (low-speed external) oscillator (32 kHz)

- clock from LSI (low-speed internal) oscillator (~ 32 kHz)
- clock from MSI (low-power internal) oscillator (~ 16 MHz)

The RCC offers a good flexibility for the application to select the appropriate clock for CPUs and peripherals, especially for peripherals that need a specific clock (such as SPI(I2S), SAI and SDMMC).

Each clock source can be switched on or off independently in order to optimize the power consumption.

There are mainly three clock paths:

- CPU1 bus matrix
- CPU2 and its bus matrix
- peripheral kernel clocks

Clocking of CPU1 and DDR interfaces are derived locally because of high frequency use. The RCC manages only the source clocks for their related local PLLs.

The RCC provides a flexible clock engine named flexclkgen based on five PLLs and HSI, HSE, MSI and CK_IN clocks to generate all root clocks.

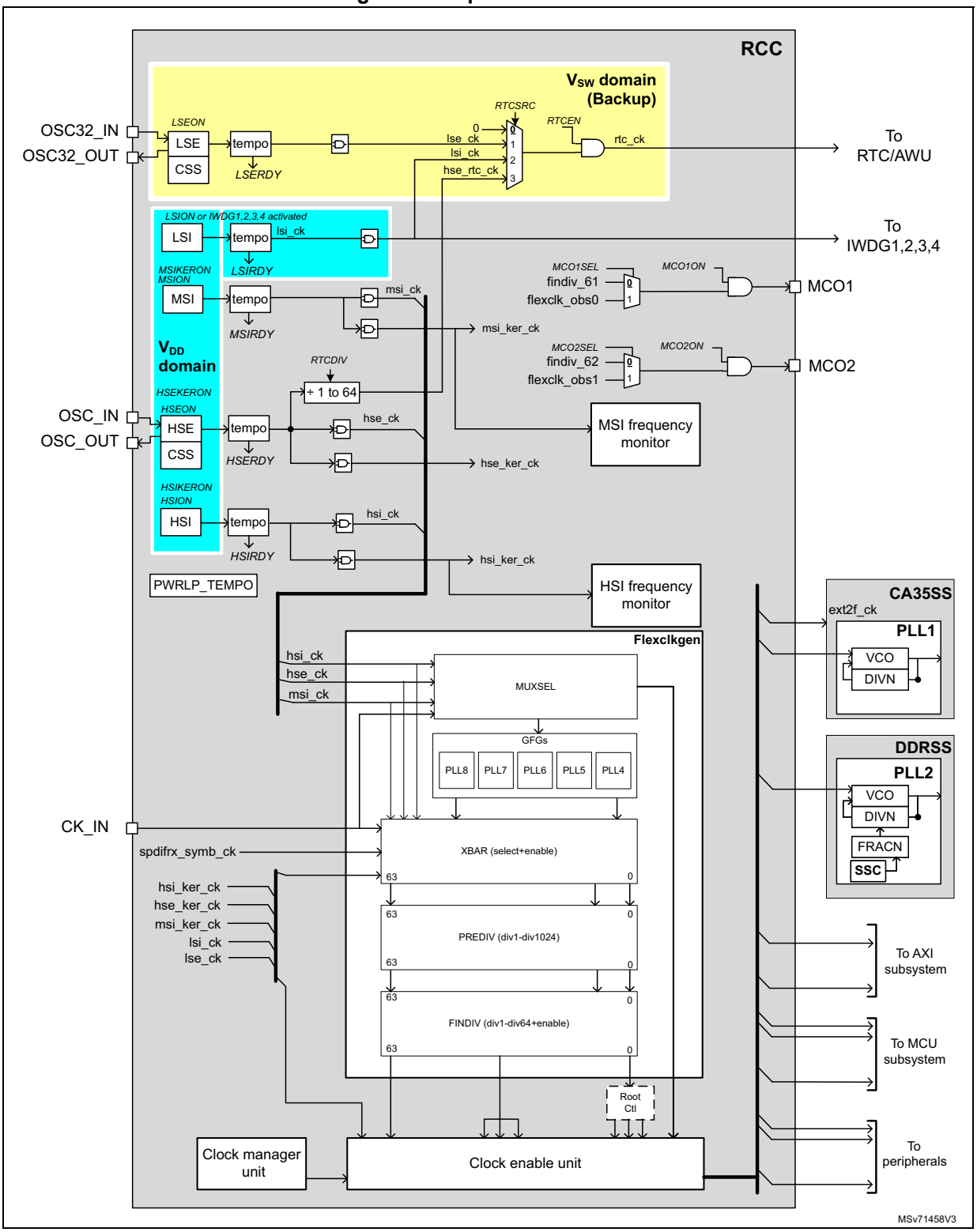
These root clocks are used in the final clock gating stage to generate all product clocks according to the different product system modes (such as Run, Stop, or Standby).

The block named clock enable unit is used to gate the peripheral kernel clock, the bus interface, cores, and bus matrix clocks.

The block named clock manager unit is responsible of the clock stop and restore sequences according to the CPU power states.

As shown in *Figure 83*, the RCC offers two clock outputs (MCO1 and MCO2), with a great flexibility on the clock selection and frequency adjustment.

Figure 83. Top-level clock tree



20.4.1 Clock naming convention

The RCC provides clocks to the complete circuit. In order to avoid misunderstanding, the following terms are used in this document:

- **peripheral clocks**

  Clocks provided by the RCC to peripherals (xxx as peripheral name):

  – programing bus interface clocks ck_icn_p_xxx
  – master data bus interface clocks ck_icn_m_xxx
  – slave data bus interface clocks ck_icn_s_xxx
  – kernel clocks ck_ker_xxx

  A peripheral receives from the RCC one or several bus interface clocks. This clock is generally the AHB, APB, or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (such as GPIOx or DMAx).

  Some peripheral request as well a dedicated clock in order to handle the interface function. This clock is named kernel clock. For example peripherals such as SAI, need to generate specific and accurate master clock frequencies, which request dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific need of the interface, is that the bus clock can be changed without requesting any reprogramming of the peripheral.

- **external PLL reference clocks**

  For CPU1 and DDR, the RCC provides the following clocks:

  – ck_pll1_ref source clock for PLL1 (CPU1)
  – ck_pll2_ref source clock for PLL2 (DDR)

- **bus matrix clocks**

  Clocks provided to the different bridges (APB, AHB or AXI), called ck_icn_xxx, with xxx is the bus matrix island clock name

20.4.2 Oscillator description

The table hereafter shows the oscillator states versus system modes when the oscillators are enabled via registers. The term “available” means that the resource can be used if activated via registers.

Table 120. Oscillator states versus system modes

System modes	V_{DD} domain					V_{SW} domain
	HSE	HSI	RCBSEC	LSI	MSI	LSE
Exit from system reset	OFF	ON	ON	Available	OFF	Available
In Run1, Run2	Available	Available	ON	Available	Available	Available
In LP-Stop, Stop1, Stop2	Available(1)	Available(2)	OFF	Available	Available(3)	Available
In LPLV-Stop	OFF	OFF	OFF	Available	OFF	Available
In Standby	OFF	OFF	OFF	Available	OFF	Available
In V_{BAT}	OFF	OFF	OFF	OFF	OFF	Available

1. HSE can remain activated in Stop mode if HSEKERON = 1, or if the peripheral generates a kernel clock request.

2. HSI can remain activated in Stop mode if HSIKERON = 1, or HSION = 1, or if the peripheral generates a kernel clock request or a bus interface clock request (autonomous peripheral).
3. MSI can remain activated in Stop mode if MSIKERON = 1, or if the peripheral generates a kernel clock request.

HSE oscillator

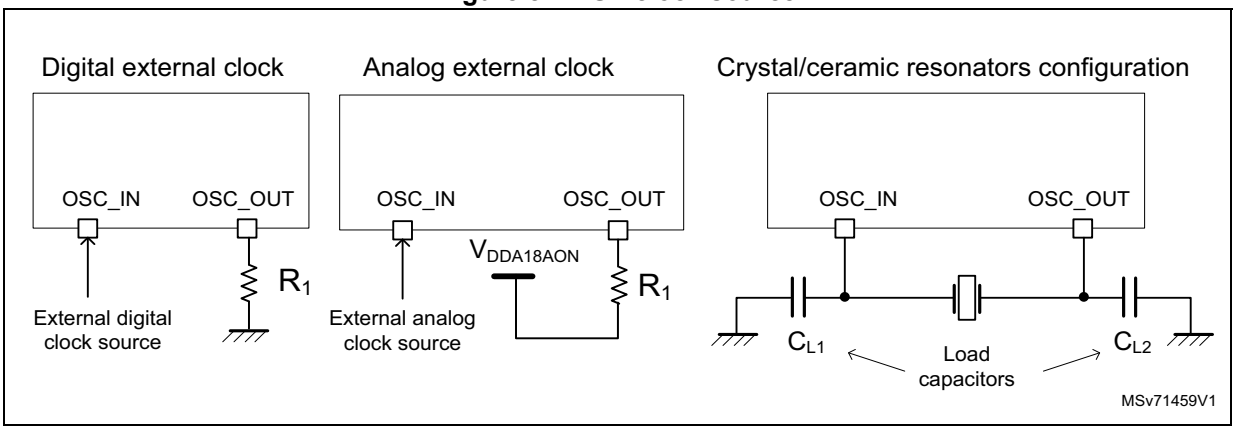
The HSE can generate a clock from three possible sources:

- external analog clock source
- external digital clock source
- external crystal/ceramic resonator

In order to allow the boot ROM to detect in which configuration the HSE is used, a resistor (R_1) must be connected either to the GND or to V_{DD}.

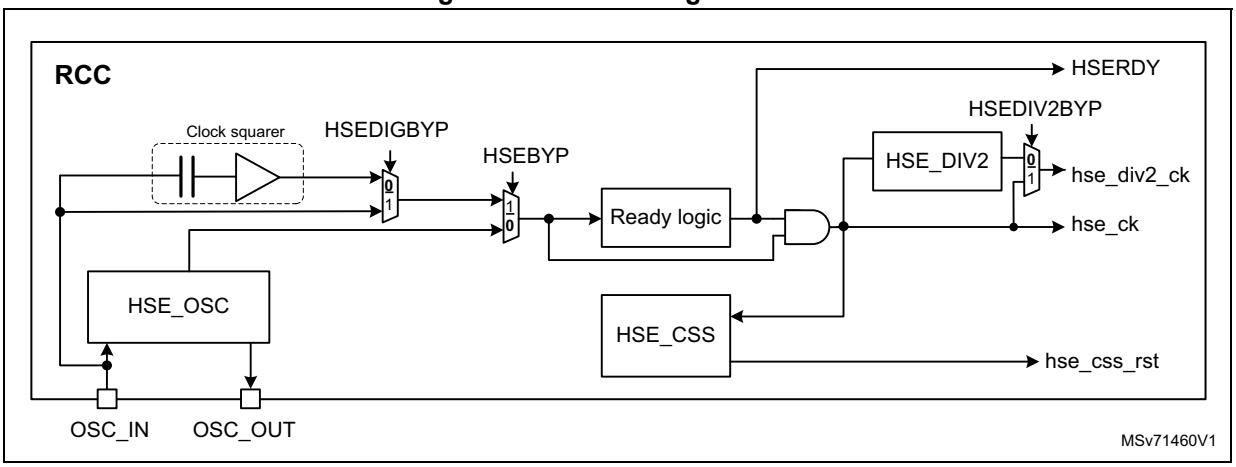
The resistor must be connected to GND when the HSE uses an external digital clock. The resistor must be connected to $V_{DDA18AON}$ when the HSE uses an external analog clock, and removed if a crystal or ceramic resonator is used. See the datasheet for C_{L1}, C_{L2} and R_1 values.

Figure 84. HSE clock source



HSERDY in RCC_OCRDYR indicates if the HSE is stable or not. When the HSE is enabled (HSEON = 1), HSERDY goes to 1 when 1024 cycles of HSE clock have been counted. The hse_ck clock is not released until HSERDY is set by hardware. An interrupt can be generated if enabled in RCC_C1CIESETR or RCC_C2CIESETR.

Figure 85. HSE clock generation



The HSE frequency can also be measured via the clock frequency calculator module in the flexclkgen engine (see *Section 20.6.1* for details).

Note: The HSE can be kept enabled when the system is in one of LP-Stop or Stop modes (see Section 20.4.8).

The HSE clock can be driven to MCO1 and MCO2 outputs, and can be used as a clock source for other application components.

External clock configuration (HSE bypass)

The HSE accepts an external clock source on OSC_IN when the bypass mode is selected (HSEBYP = 1).

The external clock signal connected to OSC_IN pin can be digital or low-swing (analog). The low-swing clock signal is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%.

In order to switch the HSE in external clock mode, the following steps are needed:

1. Disable the HSE by writing 1 to HSEON in RCC_OCENCLRR.
2. Check that the HSE is disabled (HSERDY = 0 in RCC_OCRDYR).
3. Set HSEDIGBYP
   - to 1 if the clock provided to OSC_IN is full-swing digital signal
   - to 0 if the clock provided to OSC_IN is low-swing signal

   This bit is controlled via RCC_OCENSETR and RCC_OCENCLRR.
4. Set HSEBYP bit by writing 1 to HSEBYP in RCC_OCENSETR.
5. Enable again the HSE by writing 1 to HSEON in RCC_OCENSETR, to activate hse_ck.
6. Check that the HSE is ready for use (HSERDY = 1 in RCC_OCRDYR).

The OSC_OUT pin must be left HI-Z (see *Figure 84* and *Figure 85* for details). Refer to the datasheet for additional information.

External crystal/ceramic resonator configuration

The oscillator is enabled by setting HSEBYP to 0 and HSEON to 1.

The HSE can be used when the product needs a very-accurate high-speed clock. When HSEDIV2ON = 1 in RCC_OCENSETR, a divide-by-two clock path (hse_div2_ck) is available for dedicated peripherals that require the same low noise sensitivity as USB2PHY1 or USB2PHY2. HSEDIV2BYP in RCC_OCENSETR selects the hse_div2_ck clock source.

The associated hardware configuration is shown in *Figure 84*. The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize the output distortion and the startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator (see the electrical characteristics of the datasheet for more details).

HSI oscillator

The HSI provides the default clock to the product. It is a high-speed internal RC oscillator which can also be used directly as clock source for CPUx, interconnects, and peripherals, or as PLL input.

Some HSI advantages are listed below:

- no need of external crystal (low-cost clock source)
- startup time faster than HSE (few microseconds)
- reduced power consumption

The HSI frequency precision, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

Care must be taken when the HSI is used as kernel clock for communication peripherals: the application has to take into account the following parameters:

- time interval between the moment where the peripheral generates a kernel clock request, and the moment where the clock is really available
- frequency accuracy

Note: The HSI can be kept enabled when the system is in one of LP-Stop or Stop modes (see Section 20.4.8).

HSIRDY in RCC_OCRDYR indicates if the HSI is stable or not. At startup, the HSI output clock is not released until HSIRDY is set by hardware. The HSI activation can also be controlled by the application via HSION in RCC_OCENSETR and RCC_OCENCLRR.

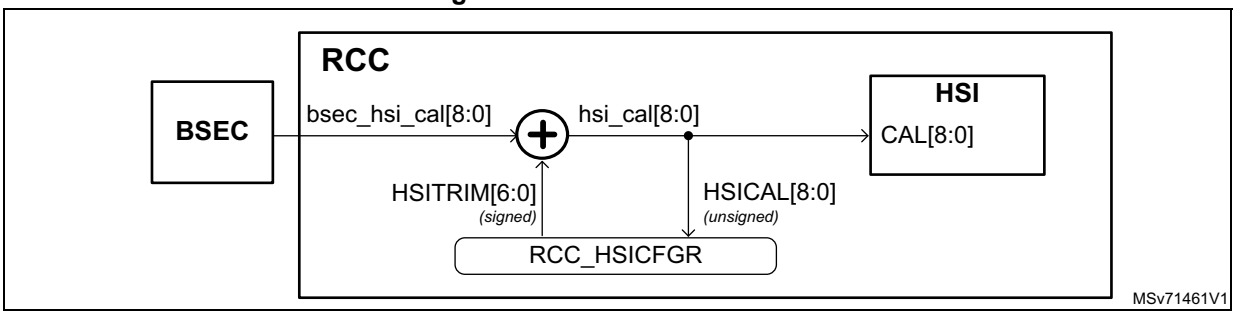
The HSI clock can be driven to MCO1 and MCO2 outputs and can be used as a clock source for other application components.

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. This is why each device is factory calibrated by ST for an accuracy of ACC_{HSI} (see the product datasheet for more information).

When an app_rstn occurred, the factory calibration value is reloaded in the HSI, and is visible in HSICAL[8:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user can trim the HSI frequency using the HSITRIM[6:0] bits. HSICAL[8:0] and HSITRIM[6:0] are in RCC_HSICFGR (see *Section 20.6.2* for additional information).

Figure 86. HSI calibration flow



Note: The HSI clock divided by 8 is also used for I/O compensation mechanism, and needs to be enabled if the I/O compensation mechanism is activated (see Section 23: System configuration controller (SYSCFG) for details).

MSI oscillator

The MSI is a low-power RC oscillator which can also be used directly as source clock for CPUx, interconnects, peripherals, or as PLL input.

The following point must be considered: If the MSI clock is currently used as kernel clock for some peripherals, the application has to ensure that the MSI frequency change does not disturb the peripherals.

Some MSI advantages are listed below:

- no need of external crystal (low-cost clock source)
- startup time faster than HSE (few microseconds)
- very-low-power consumption

Note: The MSI provides a clock frequency of 16 MHz, while the HSI is able to provide a clock up to 64 MHz.

The MSI frequency precision, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

Even if the MSI settling time is faster than the HSI one, care must be taken when the MSI is used as kernel clock for communication peripherals: the application has to take into account the following parameters:

- time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- frequency accuracy

Note: The MSI can be kept enabled when the system is in one of LP-Stop, Stop modes (see Section 20.4.8 for additional information).

The MSI clock can be driven to MCO1 and MCO2 outputs and can be used as a clock source for other application components.

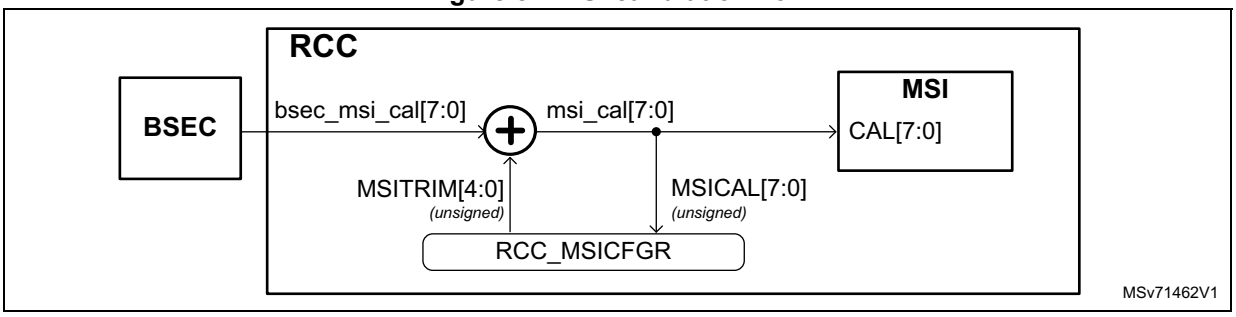
The MSI can also be controlled by the application via MSION in RCC_OCENSETR and RCC_OCENCLRR.

MSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. This is why each device is factory calibrated by ST for an accuracy of ACC_{MSI} (see the product datasheet for more information).

When an app_rstn occurred, the factory calibration value is reloaded into the MSI, and is visible on MSICAL[7:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user can trim the MSI frequency using the MSITRIM[4:0] bits. MSICAL[7:0] and MSITRIM[4:0] are in RCC_MSICFGR (see *Section 20.6.2* for additional information.

Figure 87. MSI calibration flow



RCBSEC oscillator

The RCBSEC is a 64 MHz RC oscillator dedicated to clock the BSEC.It is always activated after a vcore_rstn reset.

When the system goes in Stop or Standby mode, the RCBSEC clock is disabled by hardware (see *Section 20.4.8* for additional information).

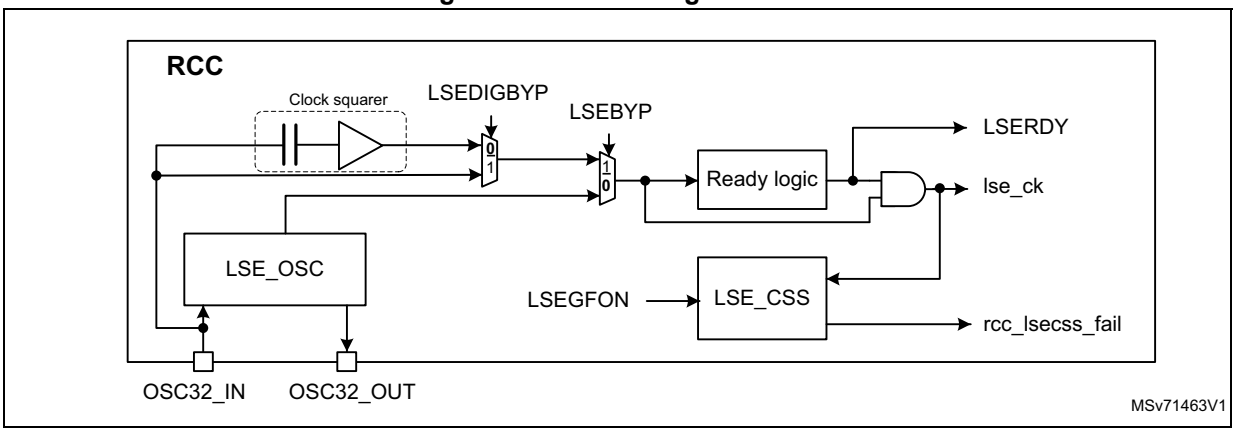
Note: The RCBSEC is also used as the source of the RNG1 and RNG2 kernel clocks. Peripheral kernel clock gating rules apply.

LSE oscillator

The LSE can generate a clock from two possible sources: external crystal/ceramic resonator, or external user clock.

The LSE offers a LSERDY flag which indicates if the LSE clock is available or not. When the LSE is enabled (LSEON = 1), LSERDY goes to 1 when 4096 cycles of LSE clock have been detected. This mechanism is also activated for an external clock source (LSEBYP = 1) with LSERDY set after the detection of 16 cycles of LSE clock. An interrupt can be generated if enabled in RCC_C1CIESETR or RCC_C2CIESETR when the LSE is ready.

Figure 88. LSE clock generation



The LSE oscillator is switched on and off using the LSEON bit. The LSE remains enabled when the system enters Stop, LP-Stop, LPLV-Stop, Standby, or V_{BAT} mode.

The LSE clock can be driven to MCO1 and MCO2 outputs and can be used as a clock source for other application components.

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 32.768 kHz, and be low-swing. This mode is selected by setting LSEBYP to 1.

The external clock signal can be digital or analog (square, sinus or triangle). Analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. The OSC32_OUT pin must be left HI-Z (see *Figure 84* and *Figure 85* for details). Refer to the datasheet for additional information.

In order to select the external clock source, the following steps are needed:

1. Set DBP = 1 in PWR_BDCR to allow write accesses to RCC_BDCR.
2. Disable the LSE by writing 0 to LSEON.
3. Check that the LSE is disabled (LSERDY = 0).
4. Set LSEDIGBYP
   - to 1 if the clock provided to OSC32_IN is full-swing digital signal
   - to 0 if the clock provided to OSC32_IN is low-swing signal
5. Set LSEBYP to 1.
6. Enable again the LSE by writing 1 to LSEON.
7. Check that the LSE is ready for use (LSERDY = 1).
8. Set DBP = 0 in PWR_BDCR to write-protect accesses to RCC_BDCR.

Note: All these bits (except DBP*) are in RCC_BDCR.*

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the RTC for clock/calendar or other timing functions.

In order to select the external clock source, the following steps are needed:

- Set DBP = 1 in PWR_BDCR to allow write accesses to RCC_BDCR.
- Disable the LSE by writing 0 to LSEON.
- Check that the LSE is disabled (LSERDY = 0).
- Clear LSEBYP to 0.
- Enable again the LSE by writing 1 to LSEON.
- Check that the LSE is ready for use (LSERDY = 1.
- Set DBP = 0 in PWR_BDCR to write-protect accesses to RCC_BDCR.

Note: All these bits (except DBP*) are in RCC_BDCR.*

The LSE on MCO1 or MCO2 outputs is only available in Run, LP-Stop, or Stop. In LPLV-Stop, Standby and V_{BAT}, LSE is not available from MCO1 or MCO2 outputs. When entering Standby or V_{BAT}, a spike may occur on MCO1 or MCO2.

The LSE offers a programmable driving capability (LSEDRV[1:0]) of the oscillator amplifier. The driving capability must be changed before enabling the LSE oscillator.

Warning: It is not allowed to change the driving capability when the LSE is enabled. The LSE behavior is not guaranteed in that case.

LSI oscillator

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode, for the independent watchdogs (IWDGx) and auto-wake-up unit (AWU). The clock frequency is around 32 kHz (see electrical characteristics in the product the datasheet for details).

The LSI can be switched on and off using LSION in RCC_LSICR. LSIRDY in RCC_LSICR indicates if the LSI oscillator is stable or not. If an independent watchdog is started by either hardware option or software access, the LSI is forced on and cannot be disabled.

At LSI startup, the clock is not provided until the hardware sets LSIRDY. An interrupt can be generated if enabled in RCC_C1CIESETR or RCC_C2CIESETR.

The LSI clock can be driven to MCO1 and MCO2 outputs and can be used as a clock source for other application components.

Note: The LSI on MCO1 or MCO2 outputs is only available in Run, LP-Stop or Stop. In LPLV-Stop and Standby, LSI is not available from MCO1 or MCO2 outputs. When entering Standby, a spike may occur on MCO1 or MCO2.

20.4.3 Clock security system (CSS)

CSS on HSE

The RCC features a clock security system allowing the application to detect if the HSE clock stops toggling due to an unexpected reason, or if the HSE frequency clock is below 8 MHz. This may be caused for example, by an hardware failure or an attack. In the case of failure detection, the CSS generates a system reset, and protects the sensitive BKPSRAM data.

The CSS can be activated by the application software via HSECSSON. This bit can be activated even when HSEON = 0. The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON = 1.

The CSS on HSE is disabled when the HSE is disabled (system is Stop or Standby mode).

HSECSSON cannot be cleared by software. It is cleared by hardware when a system reset occurs, or when the system enters Standby mode.

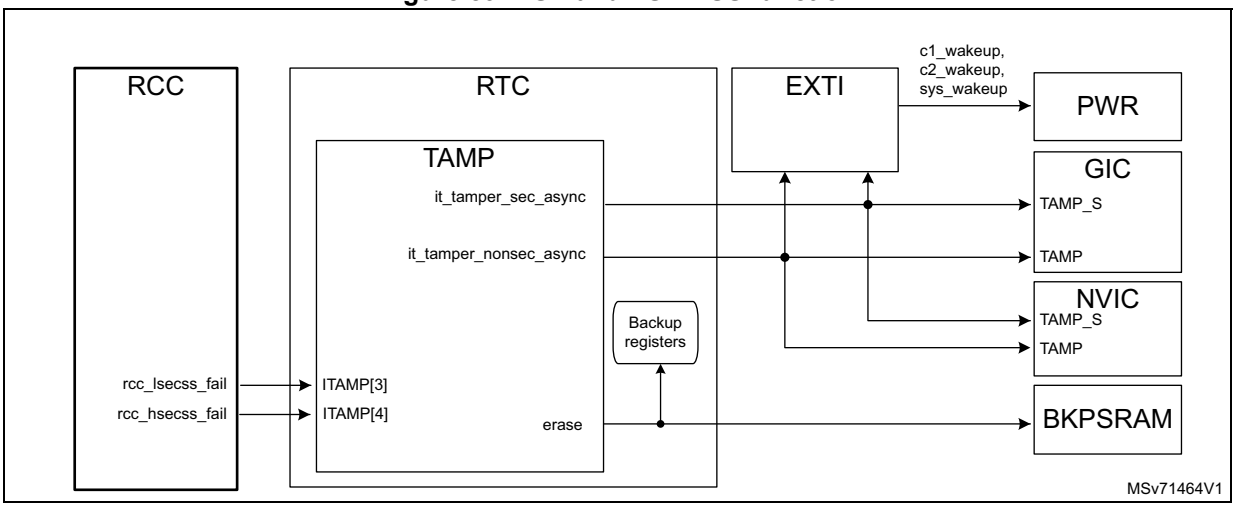
If a failure is detected on the HSE:

- RCC generates an application reset (app_rstn) via hse_css_rst. HCSSRSTF is set in order to allow the application to identify the reset root cause.
- A failure event (rcc_hsecss_fail) is generated. This event is connected to the TAMP block, allowing the protection of backup registers and BKPSRAM.

Figure 89 shows the blocks involved in the HSE and LSE CSS function. Refer to these blocks description for more details.

Note: Before selecting again the HSE clock after an HSE failure, HCSSRSTF must be cleared in RCC_HWRSTSCLRR. The Boot ROM performs this task by resetting HCSSRSTF after an HSE failure. The Boot ROM reboots then the circuit if the selected boot source does not require the HSE resource. Refer to Boot ROM documentation for details.

Figure 89. HSE and LSE CSS function



CSS on LSE

The RCC features a CSS allowing the application to detect if the LSE clock stops toggling due to an unexpected reason, or if the LSE frequency clock is above 2 MHz max frequency. This may be caused for example, by an hardware failure or an attack. In case of failure detection, the CSS generates a failure event, disables the clock provided to the RTC, and protects sensitive BKPSRAM data.

The CSS on LSE can be activated by setting to 1 LSECSSON in RCC_BDCR. This bit can be disabled only by hardware in one of the following conditions:

- after a vsw_rstn reset (see *Figure 74*)
- after a failure detection on LSE

The CSS on LSE activation must be done in the following order:

1. Set LSEGFON to 1 in RCC_BDCR to filter glitch on LSE clock.
2. Set LSEON to 1 in RCC_BDCR, and wait for LSERDY = 1 in RCC_BDCR.
3. Select the LSE clock via RTCSRC in RCC_BDCR.
4. Set LSECSSON to 1 in RCC_BDCR.

The CSS on LSE works in all system modes.

If a failure is detected on the LSE:

- The clock provided to the RTC is disabled immediately by the hardware (with RTCSRC = 0).
- LSEGFON, LSEON and LSECSSON values are not changed by hardware.
- A failure event is generated (rcc_lsecss_fail). This event is connected to the TAMP, allowing wake-up from Standby, and protection of backup registers and BKPSRAM.
- LSECSSF interrupt flag is activated in order to generate an interrupt to the CPU1 or CPU2 if needed.

The CSS on LSE offers two flags in order to detect a failure: LSECSSD in RCC_BDCR, and LSECSSF in RCC_C1CIFCLRR and RCC_C2CIFCLRR.

In case of failure, the following steps are needed:

1. Disable the CSS function (mandatory).
   a) Clear LSECSSON to 0.
   b) Clear LSEON to 0 in order to disable the LSE.
   c) Clear LSECSSF if interrupt was enabled for this event.
2. Change the clock source for the RTC if needed:
   a) Clear RTCEN to 0 to disable the RTC clock.
   b) Enable the new clock source for the RTC.
   c) Select the proper clock source via RTCSRC.
   d) Set RTCEN to 1 to enable the RTC clock.

The software can take any other required actions to secure the application.

Figure 89 shows the blocks involved in the HSE and LSE CSS function.

Note: The clock security feature must not be re-enabled.

Before reusing the CSS function for LSE, a backup domain reset must be performed for example via VSWRST in RCC_BDCR.

20.4.4 Clock output generation (MCO1 and MCO2)

Two clock output (MCO) pins are available: MCO1 and MCO2. A clock source can be selected for each one. MCO1 and MCO2 outputs are provided via two channels of the flexclkgen engine (see *Figure 90* for additional information).

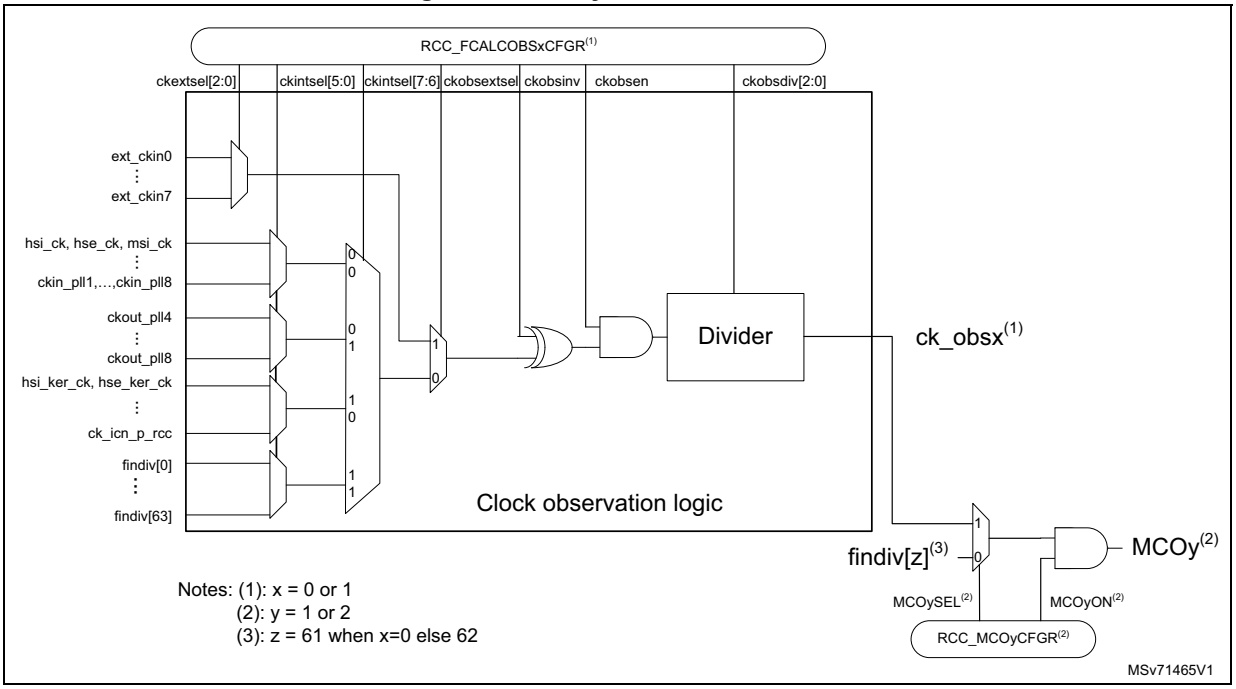
MCO1 and MCO2 outputs are controlled via MCO1ON, MCO1SEL, MCO2ON and MCO2SEL in RCC_MCO1CFGR and RCC_MCO2CFGR.

For each MCO pin, the corresponding GPIO port has to be programmed in the right alternate function mode. When the final divider channel is used for these outputs, the duty cycle is 50%.

MCO1 and MCO2 outputs are only available in Run and Stop modes. In Stop mode, the PLL clocks are not available.

Caution: The clock provided to the MCOs outputs must not exceed the maximum I/O speed (see the product datasheet for information about the supported I/O speed).

Figure 90. MCOy clock observation



20.4.5 PLL description

The RCC features five PLLs with the same features:

- PLL4 and PLL5 used to provide the system clocks
- PLL6 used for the generation of kernel clocks to video peripherals.
- PLL7 and PLL8 used for the generation of kernel clocks to audio peripherals.

Note: The above PLL use is only a recommendation.

PLLTS16FFCLAFRAC2 main features

- VCO range between 800 and 3200 MHz
- Input frequency range:
  - Integer mode: 5 MHz to 1200 MHz
  - Fractional mode: 10 MHz to 1200 MHz

  Note: The maximum input frequency is 64 MHz on the STM32MP21xx.
- Working modes
  - Fractional mode, using 24-bit delta-sigma modulator. The delta-sigma modulator can be updated on-the-fly, without generating frequency overshoots on PLL outputs.
  - Integer mode
  - Spread spectrum mode in order to reduce EMI. The fully digital component (spread-spectrum modulator called SSMOD) is used in that case.
- Up to 6 outputs:

– 1 output without post-divider
– 5 outputs with post-dividers

Figure 91. PLL block diagram

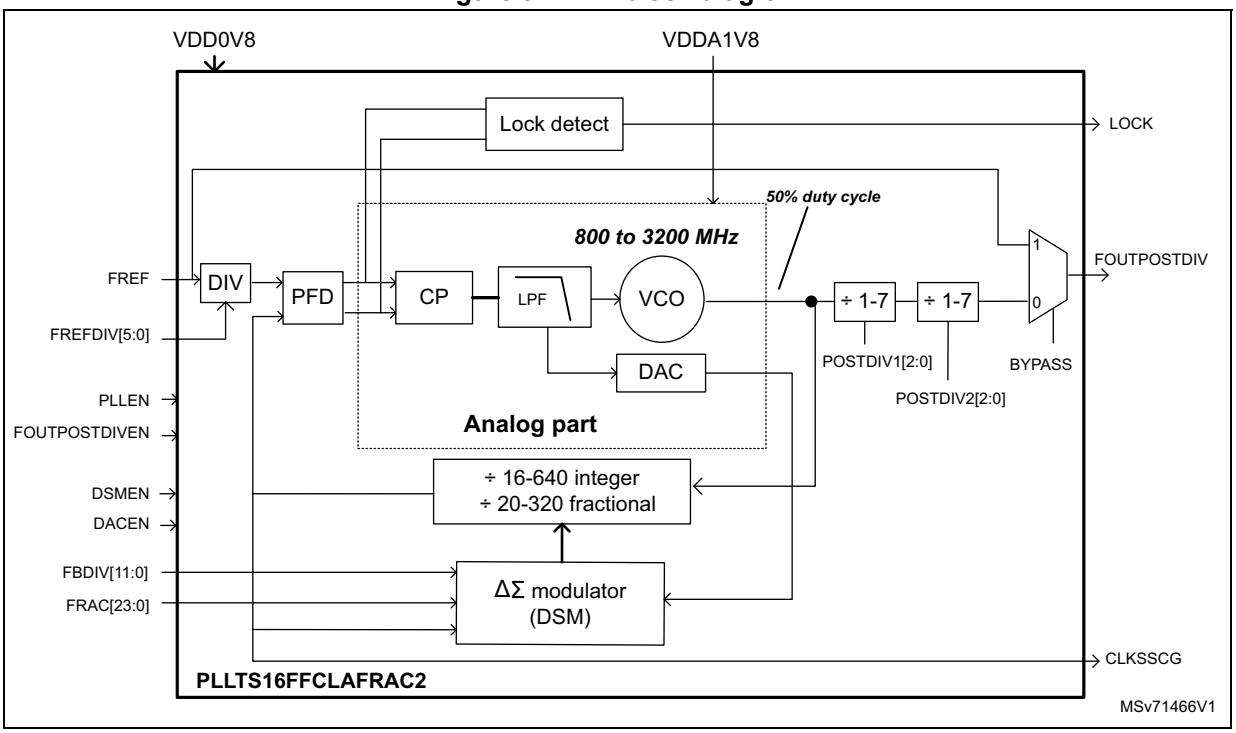
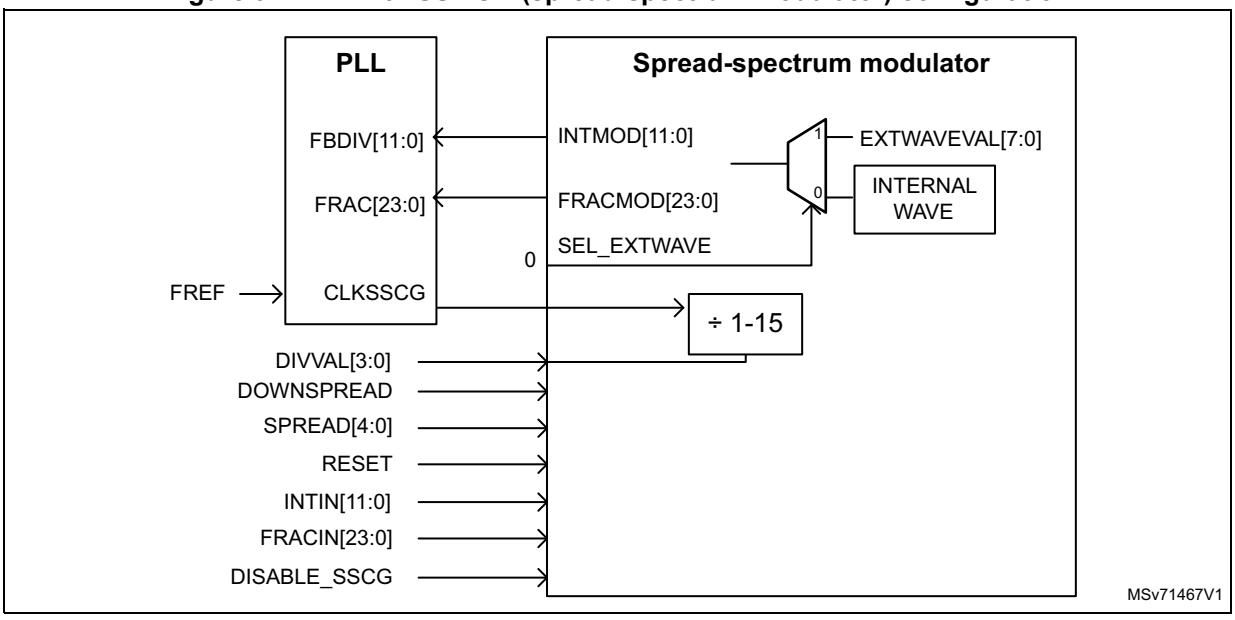


Figure 92. PLL with SSMOD (spread-spectrum modulator) configuration



The FOUTPOSTDIV output is provided as a source clock on the flexclkgen crossbar stage. The PLLs are embedded in the generic frequency generator (GFG) stage of the flexclkgen component, and are controlled via RCC_PLLxCFGRy (x = 4 to 8 and y = 1 to 7).

The user must provide to each PLL, a clean reference clock (FREF) in the accepted input frequency range:

- 5 to 1200 MHz in the integer mode
- 10 to 1200 MHz in the fractional mode

The application has to program properly the FREFDIV dividers in order to match these conditions. The PLL loop divider (FBDIV) has to be programmed in order to get the expected frequency at VCO output. The VCO output range must be respected.

The PLLs can be disabled by setting the corresponding PLLEN bit to 0. LOCK bits inform the user that the PLL is ready (locked).

When the PLLs are enabled, post-dividers (POSTDIV1, POSTDIV2) values can be changed without disabling the corresponding PLL.

PLL programming recommendations

This section provides recommendations to follow in order to guarantee a good PLL use.

- Before enabling the PLLs, the user must ensure that the reference frequency (FREF) provided to the PLL is in the good range.
- When the PLLs are enabled, changing FREFDIV, FBDIV, FRAC, POSTDIV1,or POSTDIV2 values conducts to an invalid output frequency, with the risk of crashing the system. There is no hardware protection. It is up to the application to avoid this situation.
- When one or several PLL outputs are not used, the application must set the corresponding enable bit PLLEN to 0.

The PLLs can work in three different modes:

- Integer mode

  The PLL works in integer mode when the delta-sigma modulator (DSM) is loaded with 0, and DISABLE_SSCG = 1.

  In order to load 0 into the DSM, the user has to perform the following sequence:

  a) Write 0 to FRAC, DSMEN, and DACEN.

  b) Write 1 to RESET.

  c) Write 1 to PLLEN.

  The VCO frequency (FOUT) is given by the following expression:

$$F_{FOUT} = \frac{F_{FREF} \times FBDIV}{FREFDIV}$$

  The frequency at the output of the post-divider is given by:

$$F_{FOUTPOSTDIV} = \frac{F_{FOUT}}{POSTDIV1 \times POSTDIV2}$$

- Fractional mode

  The fractional mode is activated when the value loaded into the DSM is different from 0, DISABLE_SSCG = 1, and DSMEN = 1.

  The DSM value can be updated at anytime by the application by executing the following sequence:

  a) Write into FRAC the new fractional value.

  b) The new FRAC value is loaded into the DSM when CLKSSCG goes from 0 to 1.

  The DSM is designed to minimize the jitter impact while allowing very small frequency steps. When the PLL is used in fractional mode, the FBDIV divider value must be initialized before enabling the PLL (FBDIV min = 20), but the FRAC value can be changed on-the-fly without disturbing the PLL output. This feature can be used either to generate a specific frequency, with a good accuracy from any crystal value, or to perform a fine tuning of the frequency on-the-fly.

  The VCO frequency is given by the following expression:

$$F_{FOUT} = \frac{F_{FREF} \times \left(FBDIV + \frac{FRAC}{2^{24}} \right)}{FREFDIV}$$

  See FOUTPOSTDIV frequency formula above.

- Spread-spectrum mode

  This mode is activated when the DSM is loaded with 0, and when DISABLE_SSCG = 0. This feature is available on all PLLs.

  The spread-spectrum technique consists of modulating the VCO frequency with a low-frequency signal (in our case a triangular signal), in order to spread the clock energy into a wider frequency band. The amount of emitted EMI peaks is then reduced.

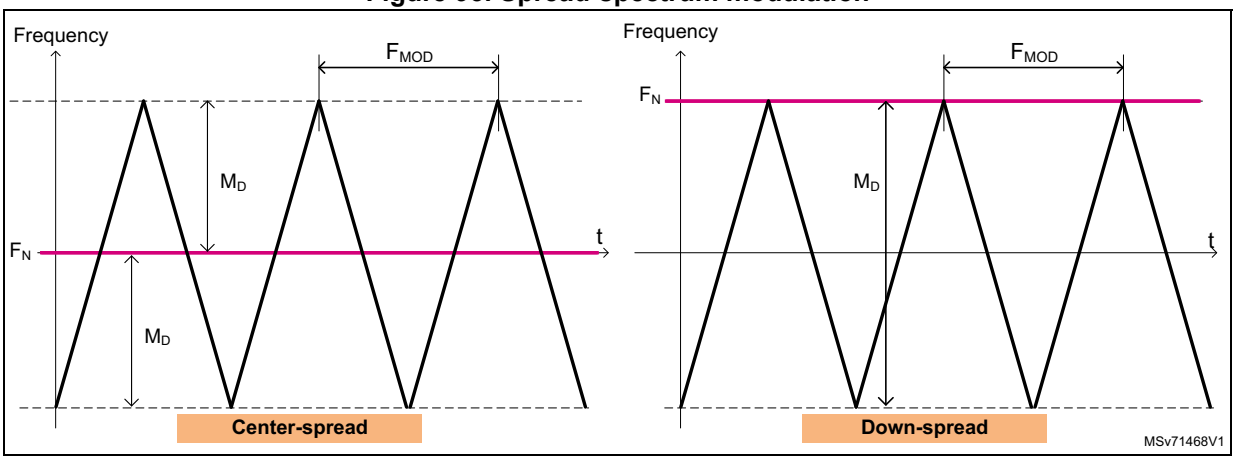
  The spread-spectrum modulation parameters are adjusted via the following fields:

  – DIVVAL[3:0] in RCC_PLLxCFGR5 used to adjust the modulation frequency
  – SPREAD[4:0] in RCC_PLLxCFGR5 used to adjust the modulation depth (or modulation index)
  – DOWNSPREAD in RCC_PLLxCFGR3 used to define if the modulation is centered around the VCO frequency (center-spread), or if it is lowered with respect to VCO frequency (down-spread)

  The figure below shows the signal modulating the nominal frequency (F_N), when DOWNSPREAD = 0 (center-spread) and DOWNSPREAD = 1 (down-spread).
  The nominal frequency is the frequency output by the PLL in integer mode, when no clock spreading is applied.

  Setting DOWNSPREAD = (down-spread) guaranties that the PLL output frequency does not exceed the programmed frequency value when the SSCG is enabled.

Figure 93. Spread-spectrum modulation



The peak modulation depth (in percentage) is given by the following formula:

$$M_D\ (\%) = \pm \text{SPREAD} \times 0.1$$

Example: SPREAD = 0b10010 means a ±1.8% modulation amplitude in center-spread mode, and -1.8% in down-spread mode.

Note: The modulation is turned off when SPREAD = 0.

The modulation frequency (F_{MOD}) is given by:

$$F_{MOD} = \frac{F_{CLKSSCG}}{128 \times \text{DIVVAL}}$$

where 128 equals to the number of points in the internal wave table, and DIVVAL equals to the user defined divide value.

Note: When the PLL is locked, $F_{CLKSSCG} = F_{PFD}$. The upper limit of the modulation frequency (F_{MOD}) is set by the PLL bandwidth.

The PLL bandwidth limits the maximum modulation to $F_{CLKSSCG}/200$ where $F_{CLKSSCG} = F_{FREF}/FREFDIV$ or 50 MHz, whichever is lower.

In order to use the spread spectrum feature, the following steps are needed:

a) Program the PLL as detailed in *Integer mode* in order to adjust the nominal frequency (F_N) according to the targeted by the application.

b) Compute DIVVAL according to the wanted modulation frequency (F_{MOD}):

$$DIVVAL = ROUND\left(\frac{F_{CLKSSCG}}{128 \times F_{MOD}}\right)$$

c) Compute SPREAD according to the wanted modulation depth (M_D).

d) Set DOWNSPREAD according to the wanted modulation type.

e) Compute INTIN accordingly (FRACIN = 0):

$$INTIN = ROUND\left(\frac{F_N \times FREFDIV}{F_{FREF}}\right)$$

f) Clear DISABLE_SSCG, DSMEN, and DACEN to 0.

g) Assert RESET to 1 and PLLEN to 0. PLLEN must be held at 0 for 1 ?s to make sure the PLL is fully reset.

h) Set PLLEN, DSMEN and DACEN to 1 (see note below).

i) Wait until the first CLKSSCG edge, and clear RESET to 0 (can be done before PLL is locked, or anytime afterwards).

Note: After some time the PLL is locked and outputs a spread-spectrum clock. Since the spread-spectrum accuracy relies on the fractional-N capability of the PLL the fractional modulator must be operating. DSMEN and DACEN must be set to 1.

The user can check F_{MIN},F_{MAX} and F_N as follow:

– If DOWNSPREAD = 0 (centered-spread)

  F_N is given by the formula in *Integer mode*

  $F_{MIN} = F_N$ x (1 - M_D/100)

  $F_{MAX} = F_N$ x (1 + M_D/100)

– If DOWNSPREAD = 1 (down-spread)

  F_{MAX} is given by the formula in *Integer mode*

  $F_{MIN} = F_{MAX}$ x (1 - M_D/100)

  $F_N = F_{MAX}$ x (1 - M_D/100)

20.4.6 PLL root clocks

The RCC handles the following PLL root clocks:

- ck_pll1_ref for CPU1 subsystem
- ck_pll2_ref for DDR subsystem

PLL root clock selection

After a system reset, the HSI is selected for the complete system. All the PLLs are switched off, as well as MSI and HSE oscillators.

When the system is running, the user has the possibility to select the clock source to be used for all above subsystem clocks. It relies on the following clock sources: HSI, HSE, MSI and CK_IN.

20.4.7 Root clocks

The RCC handles two types of root clocks:

- Bus root clocks:
  - Interconnect subsystems: ck_icn_hs_mcu, ck_icn_ls_mcu, ck_icn_sdmmc, ck_icn_ddr, ck_icn_display, ck_icn_hsl, ck_icn_nic, ck_icn_vid
  - Peripheral bridges: ck_icn_apb1, ck_icn_apb2, ck_icn_apb3, ck_icn_apb4, ck_icn_apb5, ck_icn_apbdbg
- Kernel root clocks (see *Table 125* for details)

Root clock generation

Figure 94 shows the flexclkgen engine which is used to generate all root clocks when the system is in Run mode.

After any system reset, the flexclkgen default configuration is to output the HSI reference clock (64 MHz) on all root clocks while all PLLs are disabled.

It is up to the boot code to configure the PLLs and the clock dividers with the appropriate values.

The flexclkgen is based on the following stages:

- MUXSEL stage used to select the input clock reference for all PLLs in the GFG stage and for the external clock outputs (see RCC_MUXSELCFGR)
- GFG stage used to control all PLLs (see RCC_PLLxCFGRy (x = 4 to 8 and y = 1 to 7)
- XBAR (clock cross bar) stage used to select clock channels driven to divider stages (see RCC_XBARxCFGR)
- PREDIV (pre divider) stage used for a first level of clock division (div1, div2, div4 and div1024) (see RCC_PREDIVxCFGR, RCC_PREDIVSR1, and RCC_PREDIVSR2)
- FINDIV (final divider) stage used for the last level of clock division (div1 to div64) (see RCC_FINDIVxCFGR, RCC_FINDIVSR1, and RCC_FINDIVSR2)

Figure 94. Flexclkgen block diagram

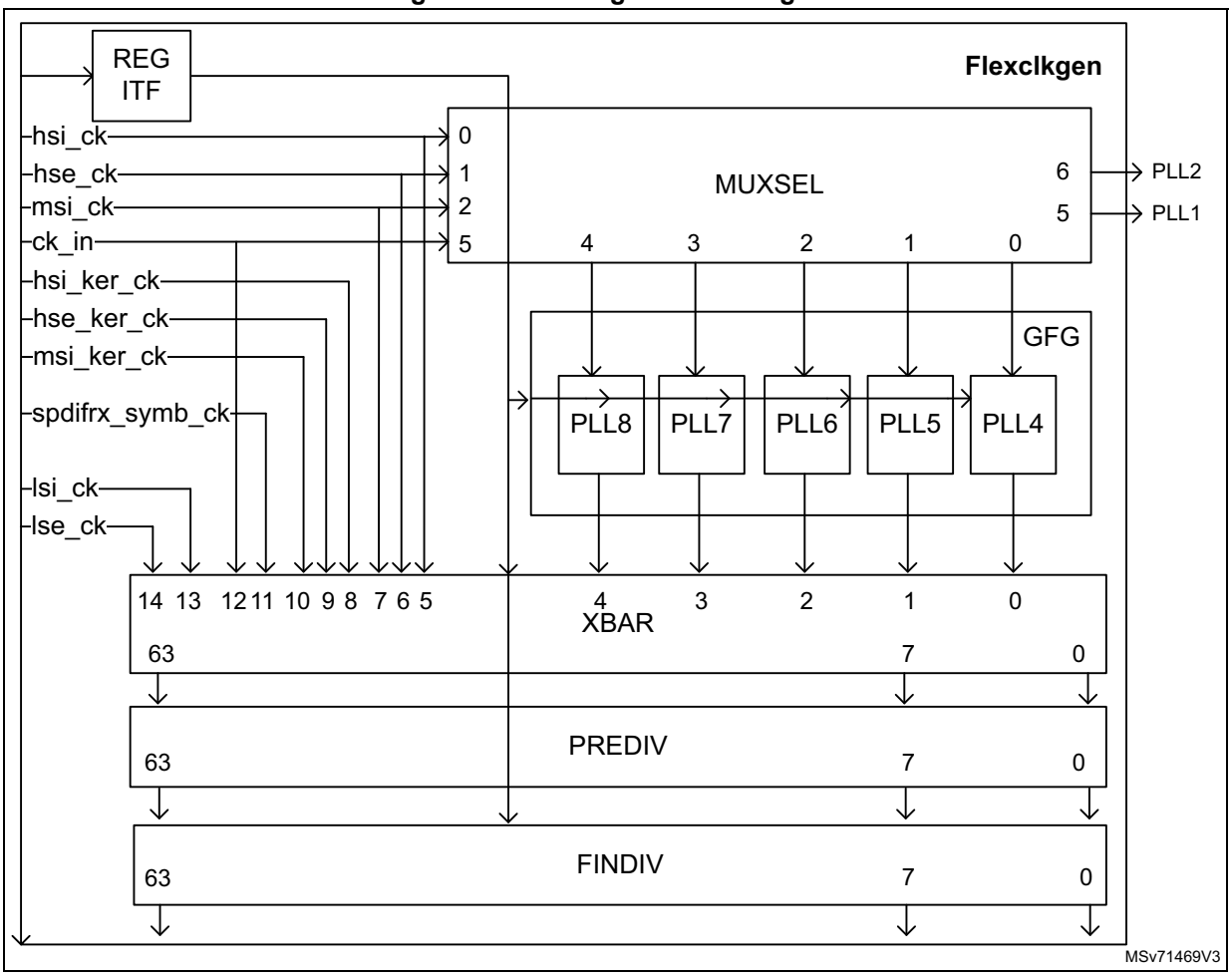
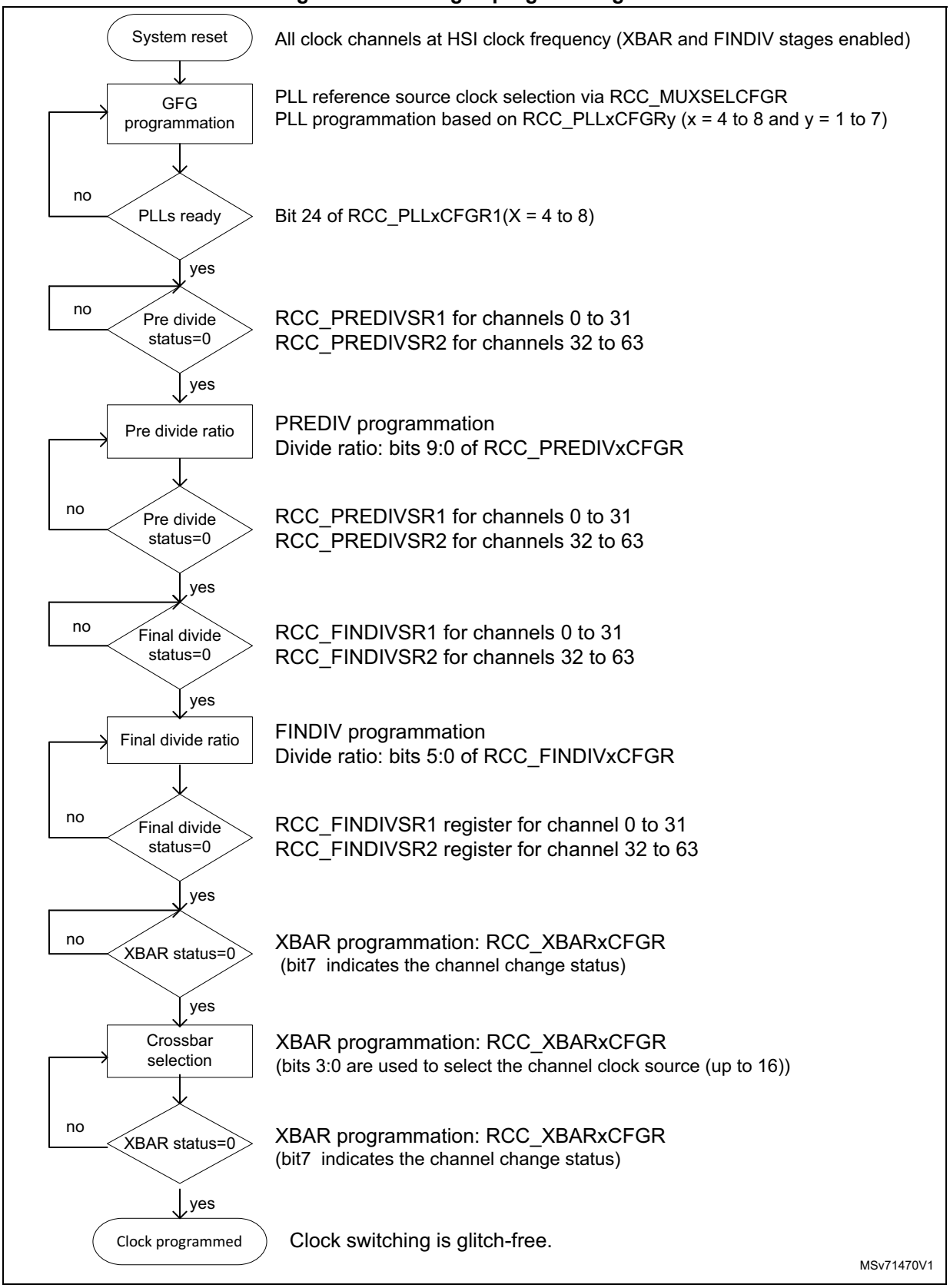


Figure 95. Flexclkgen programming flow



Caution: The channel clock used to generate ck_icn_hs_mcu and ck_icn_ls_mcu subsystem clocks cannot be stopped in Run mode as it is the source of the RCC main clock.

The table below shows the flexclkgen clock channel assignment for root clocks, and its distribution versus local peripheral bus clocks.

Table 121. Root clock mapping

Flexclkgen channel number	Root clock name	Max frequency (MHz)	Bus matrix	Target peripheral bus clock
0	ck_icn_hs_mcu	300	AXIM	ck_icn_s_fmc, ck_icn_s_sysram
			MLAHB	ck_icn_hs_mlahb, ck_cpu2, ck_cpu2_icache, ck_cpu2_dcache, ck_icn_s_retram, ck_icn_s_sram1, ck_icn_s_ospi1, ck_icn_p_otfd1
	ck_icn_ls_mcu	150	AXIM	ck_icn_m_axiap, ck_icn_s_stm
			MLAHB	ck_icn_ls_mlahb
			AHB2	ck_icn_ahb2, ck_icn_p_hpdma1, ck_icn_p_hpdma2, ck_icn_p_hpdma3, ck_icn_p_ipcc1, ck_icn_p_dcmipssi, ck_icn_p_crc, ck_icn_p_mdf1, ck_icn_p_adc1, ck_icn_p_adc2
			AHB3	ck_icn_ahb3, ck_icn_s_bkpsram, ck_icn_p_ramcfg, ck_icn_p_rifsciac, ck_icn_p_risaf1, ck_icn_p_risaf2, ck_icn_p_risaf4, ck_icn_p_risab1, ck_icn_p_risab2, ck_icn_p_risab3, ck_icn_p_risab5(1)
			AHB4	ck_icn_ahb4, ck_icn_p_rcc, ck_icn_p_pwrctrl, ck_exti, ck_icn_p_syscfg, ck_icn_p_gpioa, ck_icn_p_gpiob, ck_icn_p_gpioc, ck_icn_p_gpiod, ck_icn_p_gpioe, ck_icn_p_gpiof, ck_icn_p_gpiog, ck_icn_p_gpioh, ck_icn_p_gpioi
			AHB5	ck_icn_ahb5, ck_icn_p_fmc, ck_icn_p_eth1, ck_icn_p_eth2, ck_icn_p_syscpu1, ck_icn_p_ddrphyc
			AHB6	ck_icn_ahb6, ck_icn_p_gpioz
			APB1	ck_icn_ahb_apb1
			APB2	ck_icn_ahb_apb2
			APB3	ck_icn_ahb_apb3
			APB4	ck_icn_ahb_apb4
			APB5	ck_icn_ahb_apb5
			APBDBG	ck_icn_ahb_apbdbg
1	ck_icn_sdmmc	200	AHB5	ck_icn_p_dlybsd1, ck_icn_p_dlybsd2, ck_icn_p_dlybsd3, ck_icn_m_sdmmc1, ck_icn_m_sdmmc2, ck_icn_m_sdmmc3
2	ck_icn_ddr	600	AXIM	ck_icn_s_ddr, ck_icn_m_cpu1
3	ck_icn_display	400	AXIM	ck_icn_m_ltdc, ck_icn_m_dcmipp

Table 121. Root clock mapping (continued)

Flexclkgen channel number	Root clock name	Max frequency (MHz)	Bus matrix	Target peripheral bus clock
4	ck_icn_hsl	300	AXIM	ck_icn_m_usbhehci, ck_icn_m_usbhohci, ck_icn_m_otg, ck_icn_m_eth1, ck_icn_m_eth2
5	ck_icn_nic	400	AXIM	ck_icn_s_bootrom, ck_icn_s_gic400

1. ck_icn_p_risaf1, ck_icn_p_risaf2, ck_icn_p_risaf4, ck_icn_p_risab1, ck_icn_p_risab2, ck_icn_p_risab3, ck_icn_p_risab5 clocks are enabled according to the related resource clock gating logic. Refer to RISAB and RISAF sections for resource assignment.

The root control module shown in *Figure 96* is added between the flexclkgen engine and the clock enabling stage to generate dynamically all root clocks with synchronous relationships (interconnect, bridges and timer peripherals).

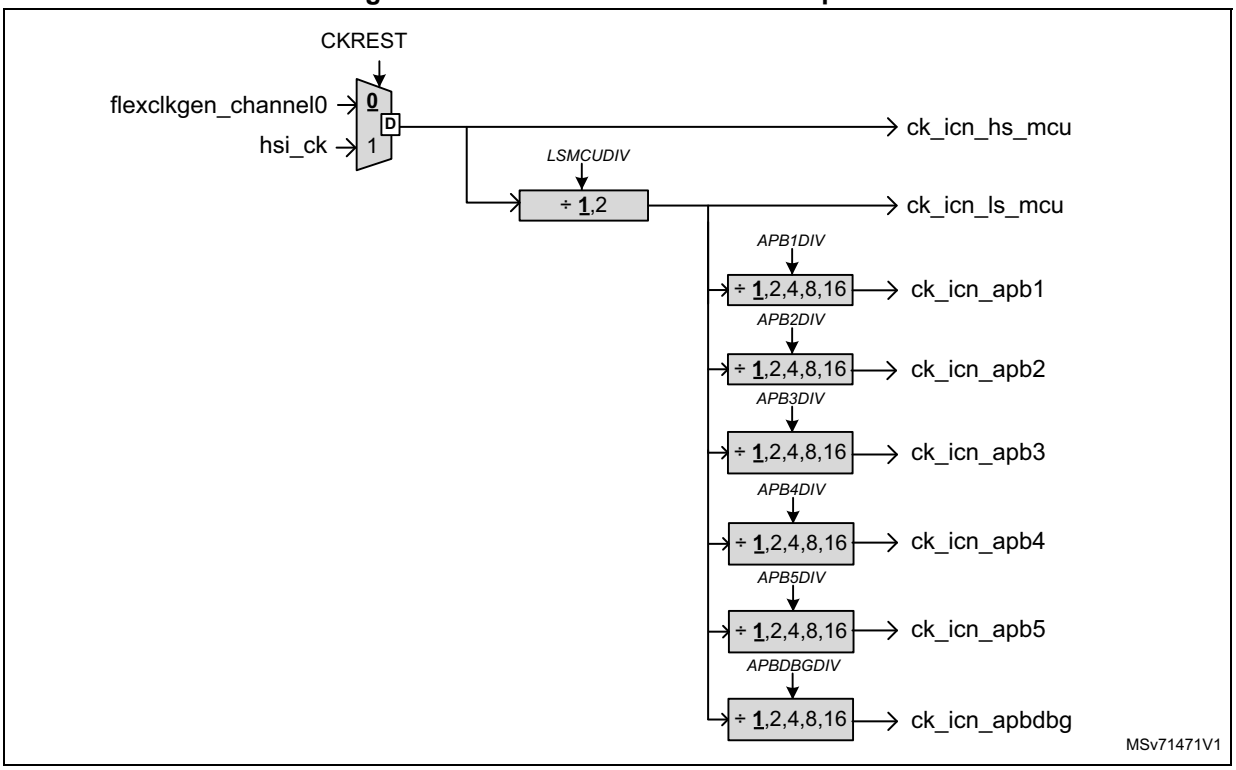
In the product, the root control module applies to the root clock ck_icn_ls_mcu, with:

- ck_icn_apb1, ck_ker_timx (x = 2 to 7, and 10 to 14)
- ck_icn_apb2, ck_ker_timy (y = 1, 8, 15, 16, 17)
- ck_icn_apb3
- ck_icn_apb4
- ck_icn_apb5
- ck_icn_apbdb

The APB1DIV divider in RCC_APB1DIVR can be used to adjust dynamically the clock for the APB1 bridge. Similar implementation is available for APB2, APB3, APB4, APB5 and APBDBG.

Note: Dividers having the possibility to be updated on-the-fly provide a ready bit in order to inform the application on the moment when the new division ratio is taken into account.

Figure 96. Root control module description.



The table below shows the peripheral root bus clock distribution versus peripherals.

Table 122. APB root clock mapping

Root clock name	Max frequency (MHz)	Target peripheral bus clock
ck_icn_apb1	150	ck_icn_p_tim2, ck_icn_p_tim3, ck_icn_p_tim4, ck_icn_p_tim5, ck_icn_p_tim6, ck_icn_p_tim7, ck_icn_p_tim12, ck_icn_p_tim13, ck_icn_p_tim14, ck_icn_p_lptim1, ck_icn_p_lptim2, ck_icn_p_spi2, ck_icn_p_spi3, ck_icn_p_spdifrx, ck_icn_p_usart2, ck_icn_p_usart3, ck_icn_p_uart4, ck_icn_p_uart5, ck_icn_p_i2c1, ck_icn_p_i2c2, ck_icn_p_i3c1, ck_icn_p_i3c2, ck_icn_p_tim10, ck_icn_p_tim11
ck_icn_apb2	150	ck_icn_p_tim1, ck_icn_p_tim8, ck_icn_p_usart6, ck_icn_p_spi1, ck_icn_p_spi4, ck_icn_p_tim15, ck_icn_p_tim16, ck_icn_p_tim17, ck_icn_p_spi5, ck_icn_p_sai1, ck_icn_p_sai2, ck_icn_p_sai3, ck_icn_p_fdcan, ck_icn_p_usart1, ck_icn_p_sai4, ck_icn_p_uart7
ck_icn_apb3	150	ck_icn_p_bsec, ck_icn_p_iwdg1, ck_icn_p_iwdg2, ck_icn_p_iwdg3, ck_icn_p_iwdg4, ck_icn_p_wwdg1, ck_icn_p_vref, ck_icn_p_dts, ck_icn_p_serc, ck_icn_p_hdp, ck_icn_p_dbgmcu
ck_icn_apb4	150	ck_icn_p_ltdc, ck_icn_p_csi, ck_icn_p_dcmipp, ck_icn_p_ddrc, ck_icn_p_ddrcfg, ck_icn_p_stgen, ck_icn_p_ddrperfm

Table 122. APB root clock mapping (continued)

Root clock name	Max frequency (MHz)	Target peripheral bus clock
ck_icn_apb5	150	ck_icn_p_rtc, ck_icn_p_spi6, ck_icn_p_lpuart1, ck_icn_p_i2c3, ck_icn_p_lptimer3, ck_icn_p_lptimer4, ck_icn_p_lptimer5, ck_icn_p_i3c3
ck_icn_apbdbg	150	ck_icn_p_soc_romtable, ck_icn_p_funnel, ck_icn_p_etf, ck_icn_p_etr, ck_icn_p_tpiu, ck_icn_p_replicator, ck_icn_p_tsgen1, ck_icn_p_stm, ck_icn_p_cti0, ck_icn_p_cti1, ck_icn_p_swo, ck_icn_p_a35_romtable, ck_icn_p_a35_dbg0, ck_icn_p_a35_cti0, ck_icn_p_a35_pmu0, ck_icn_p_a35_etm0, ck_sys_dbg

20.4.8 Clock generation in Stop and Standby modes

The product supports six system Stop modes (see *Section 19: Power control (PWR)* for details):

- Stop 1: normal system Stop mode
- LP-Stop1: low-power system Stop mode
- LPLV-Stop1: low-power, low-voltage system Stop mode
- Stop2: enhanced system Stop mode
- LP-Stop2: enhanced low-power system Stop mode
- LPLV-Stop2: enhanced low-power, low-voltage system Stop mode

There is one low-power system Standby mode.

The main difference between Stop modes is the handling of D1 and D2 domain supply voltages:

- In Stop1 mode, V_{DDCORE} and V_{DDCPU} supplies remain the same as in system Run mode, with the same power capability.
- In LP-Stop1 mode, V_{DDCORE} and V_{DDCPU} supplies remain the same as in system Run mode, but with a limited power capability. This is usually achieved thanks to an external PMIC having two power modes:
  - a high power mode (HP) for nominal cases (Run and Stop1)
  - a low-power mode (LP) limiting the power capability for LP-Stop1 and LPLV-Stop1
- In LPLV-Stop1 mode, V_{DDCORE} and V_{DDCPU} supplies are reduced, and the power capability is reduced as well.
- For all Stop2 modes, it is the same behavior as the Stop1 ones, except that V_{DDCPU} supply is powered down.
- Refer to *Section 20.4.3* for PLL1 management in Stop modes

The amount of peripherals able to wake up the system from Stop depends on the activated Stop mode. For example, Stop1, Stop2, LP-Stop1 or LP-Stop2 mode offers a larger choice of peripherals able to wake up the system than LPLV-Stop1 or LPLV-Stop2 mode.

Entering one of the Stop modes

The system can go to one these Stop modes when both CPU1 and CPU2 are in CStop without any pending wake-up event on the EXTI. To go in one of the Stop or LP-Stop modes, any autonomous peripheral that requests a bus clock delays the system entry in the requested low-power mode (see *Section 19.5.2: Low-power batch acquisition mode (LPBAM)* for the list of supported autonomous peripherals).

Once the above conditions are met, the RCC informs the PWR that CPU1 and/or CPU2 subsystems are in CStop via pwrds_cpu1, pwrds_cpu2 and pwrds_sys (see *Figure 73*). Any autonomous bus clock request is postponed until the sequence to go in Stop is done.

- pwrds_cpu1 is used to indicate to the PWR that the CPU1 is in CStop.
- pwrds_cpu2 is used to indicate to the PWR that the CPU2 is in CStop.
- pwrds_sys is used to indicate to the PWR that the system is in one of the Stop modes, and no longer clocked.

Note: When only one CPUx goes to CStop, all RCC PLLs and oscillators generating the root and kernel clocks of all shared peripherals (CPU1 and CPU2) are kept activated (see Section 20.3.11 for details on these peripherals).

The PLL1 is disabled when CPU1 is in CStop. The PLL2 state is based on DDRSHR bit and the CPU1 or system state (see *Clock distribution for the DDR* for details).

As before going to CStop, each CPUx has to mask all the interrupts except the RCC wake-up interrupt in order to insure a safe Stop sequence entry and exit. The RCC provides two wake-up interrupts (rcc_cpu1_wkup_it and rcc_cpu2_wkup_it) which can be enabled via their corresponding WKUPIE bit (see *Section 20.5* for additional information).

Note: When the D1 domain is put in DStandby mode (Stop2, LP-Stop2 or LPLV-Stop2 mode), the RCC wake-up interrupt for CPU1 is not relevant as the CPU1 reboots in case of CPU1 wake-up event.

Before switching off all the clocks, the RCC waits for the DDRCTRL to enter in self-refresh mode (SSR RCC mode is the only mode that allows the system to go in one of Stop modes). As a consequence, the effective entry in on of the Stop modes may be delayed until the DDRCTRL allows it. The RCC stops then all the clocks (subsystem and kernel clocks), and the following clock sources are disabled as well:

- HSI (if HSIKERON = 0)
- HSE (if HSEKERON = 0)
- MSI (if MSIKERON = 0)
- RCBSEC
- PLL1(CPU1), PLL2(DDR), PLL4(RCC), PLL5(RCC), PLL6(RCC), PLL7(RCC) and PLL8(RCC)

Note: In LPLV-Stop1 or LPLV-Stop2 mode, HSI, HSE and MSI cannot be used.

LSE and LSI oscillators remain active in all Stop modes if enabled.

During one of the Stop modes

As explained in the previous section, in one of the system Stop modes, all clocks, PLLs and oscillators are disabled, except LSE and LSI which are kept in their current states. There are specific cases in Stop1, Stop2, LP-Stop1 or LP-Stop2 mode, where the HSE, HSI or MSI can be enabled:

- when a dedicated peripheral requests a kernel clock

  The peripheral receives HSE, HSI, or MSI according to the kernel clock source selected for this peripheral (via RCC_XBARxCFGR).
- when HSEKERON, HSIKERON, or MSIKERON is set

  The HSE, HSI, or MSI is kept running during Stop modes but the kernel clock is not provided to the peripherals. The clock is then available immediately when the system

exits one of the Stop modes, or when a peripheral requests the kernel clock (see *Table 123* for details).

- when an autonomous peripheral requests a bus interface clock

  The provided bus interface clock is always the HSI (see *Section 19: Power control (PWR)* for the list of autonomous peripherals).

HSIKERON, MSIKERON and HSEKERON are in RCC_OCENSETR and RCC_OCENCLLR.

The table below details the behavior of these bits with 'ON' status highlighted in grey.

Note: HSE, HSI and MSI outputs provide two clock paths (see Figure 83):
- one path for the CPUx and subsystem clocks (hse_ck, hsi_ck or msi_ck)
- one path for the peripheral kernel clocks (hse_ker_ck, hsi_ker_ck or msi_ker_ck)
These paths are available via the flexclkgen engine.

Table 123. HSI, MSI and HSE states versus system modes

System mode	HSION (HSEON) (MSION)	HSIKERON (HSEKERON) (MSIKERON)	Kernel clock request(1)	Bus clock request(2)	-	HSI state (HSE state) (MSI state)	hsi_ck (hse_ck) (msi_ck)	hsi_ker_ck (hse_ker_ck) (msi_ker_ck)
Run1 or Run2	0	X	1	X	-	ON	Gated	ON
			0		-	OFF		Gated
	1		X		-	ON	ON	ON
Stop1, LP-Stop1, Stop2 or LP-Stop2	X	X	1	0	-	ON(3)	Gated	ON
		0	0		-	OFF		Gated
		1			-	ON		ON(4)
		X	X	1	-	ON	ON(5)	ON(6)
Standby	X	X	X	X	-	OFF	Gated	Gated

1. Kernel clock request is equal 1 when the peripheral requests a kernel clock, and its corresponding RCC_XBARxCFGR selects HSE, HSI or MSI, and the corresponding PERxEN and PERxLPEN are set to 1 (see *Table 140* for details).
2. Bus clock request is equal 1 when the peripheral requests a bus interface clock, and the corresponding PERxEN and PERxLPEN are set to 1, and the system is in one of the Stop modes.
3. Only the oscillator selected by RCC_XBARxCFGR of the peripheral that requests the kernel clock is enabled.
4. The kernel clock is provided to the clock enabling block, and gated as long as the peripheral does not generate a kernel clock request.
5. Only HSI is supported in these modes.
6. The kernel clock selected by RCC_XBARxCFGR is enabled.

Exiting one of the Stop modes

The system can leave one of the Stop modes thanks to events generated by peripherals able to work in the selected mode.

The exit from system Stop is generally done in the following way:

1. A peripheral sends a wake-up interrupt to the EXTI.
2. The EXTI generates an event to the PWR (sys_wakeup).
3. The PWR signals to the external PMIC or the external regulator that the main power must be restored (through PWR_LP or optionally PWR_ON/PWR_CPU_ON pins).
4. If the PMIC was in LPLV-Stop1 or LPLV-Stop2, the PWR waits for a nominal V_{DDCORE}.
5. When V_{DDCORE} reached the nominal value, the PWR asserts pwrwake_sys for the RCC.
6. Thanks to this signal, the RCC provides the hsi_ck clock to the EXTI and PWR, and starts the clock restore sequence (see *Clock restore sequence* for details).
7. When the hsi_ck is available, the EXTI asserts the cpu1_wakeup and/or cpu2_wakeup signals, indicating to the PWR which CPUx must be activated. If the PMIC was in Stop2, LP-Stop2 or LPLV-Stop2, the PWR waits for a nominal V_{DDCPU} voltage.
8. The PWR asserts the pwrwake_cpu1 signal once V_{DDCPU} has reached the nominal value when leaving Stop2, LP-Stop2 or LPLV-Stop2, and/or the pwrwake_cpu2 signal for the RCC according to the CPUx to be activated.
9. On the RCC side:
   - If pwrwake_cpu1 is activated while leaving Stop1, LP-Stop1 or LPLV-Stop1, the RCC asserts rcc_cpu1_wkup_it, and provides the clocks to the CPU1 domain when these clocks are available.
   - If pwrwake_cpu1 is activated while leaving Stop2, LP-Stop2 or LPLV-Stop2, the RCC releases the CPU1 reset, and provides the clocks to the CPU1 domain when these clocks are available.
   - If pwrwake_cpu2 is activated, the RCC asserts rcc_cpu2_wkup_it, and provides the clocks to the CPU2 domain when these clocks are available.
10. The targeted CPUx exits from CStop, and reenables the interrupts of the peripherals activated during Stop mode. The peripheral which has generated the wake-up event is then able to generate an interrupt to the CPUx.
11. The RCC acknowledges pwrwake_cpu1 or pwrwake_cpu2 requests by releasing pwrds_cpu1 or pwrds_cpu2.

The RCC integrates a state machine able to restore the clock configuration which was used before going to one of the Stop modes.

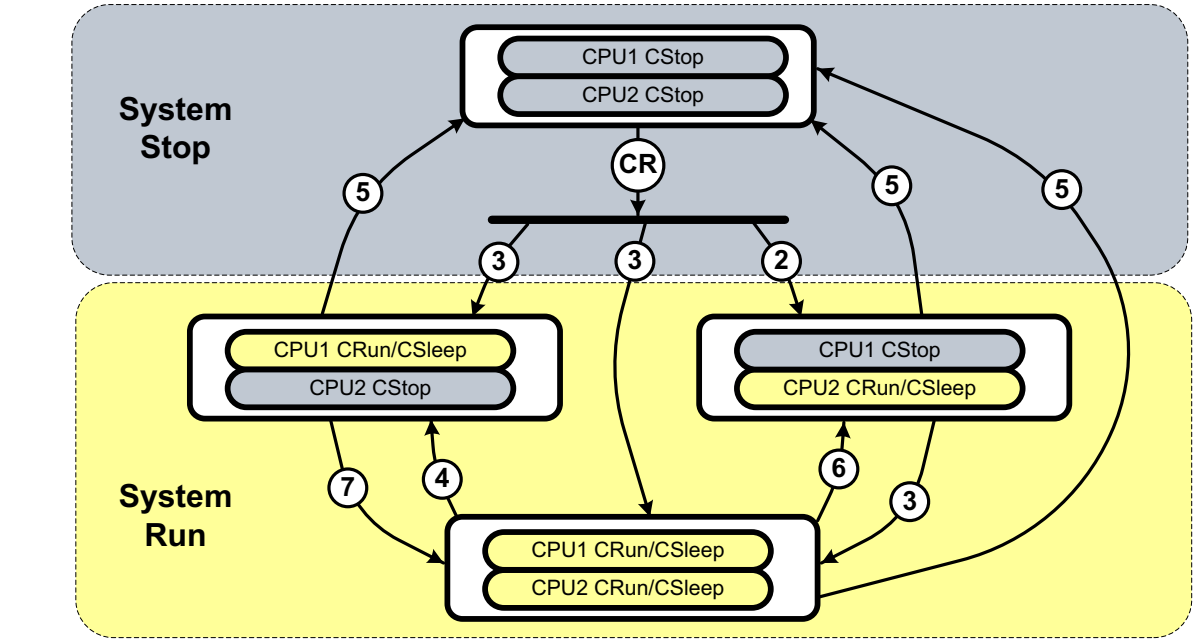
Remarks: Exiting the system from one of the Stop modes requests that peripherals are able to generate wake-up events, even when the system clock is not available.

The EXTI is used to trigger the wake-up sequence (see *Section 29: Extended interrupts and event controller (EXTI)* to get the list of peripherals that can wake-up a system from Stop or a CPUx from CStop).

The RCC executes the clock restore sequence every time the system exits Stop as shown in the figure below. When the system is in Run, all oscillators and PLLs requested by the

application are activated, but each CPU clock domain is only activated according to the related CPU mode.

Figure 97. Clock restore activation versus system mode



CR Clock restore sequence triggered by pwrwake_sys event. hsi_ck is enabled for RCC, EXTI and PWR. All PLLs are reenabled if their respective PLLxEN = 1.

2 Enabling of CPU2 clock domain (pwrwake_cpu2) once all PLLs are ready except when HSI clock is selected on CPU2

3 Enabling of ck_pll1_ref, ck_cpu1_ext2f, and CPU1 clock domain (pwrwake_cpu1)

4 Disable all clocks allocated to CPU2 domain.

5 Disable all clocks, all PLLs, and oscillators if allowed.

6 Disabling of ck_pll1_ref, ck_cpu1_ext2f, and CPU1 clock domain

7 Enable all clocks allocated to CPU2 domain.

Run, CRun or CSleep

Stop, CStop

MSv71472V1

Handling the PWRLP_TEMPO

The PWRLP_TEMPO is used to wait automatically that the external PMIC switched from low-power (LP) to high-power (HP) mode, before enabling power-consuming parts. The application can control the delay value via RCC_PWRLPDLYCR.

The PWRLP_TEMPO uses the ck_icn_ls_mcu clock as reference which is sourced during the clock restore sequence by either hsi_ck (when LSMCU = 0 in RCC_LSMCUDIVR) or by hsi_ck/2 (when LSMCU = 1).

In order to allow a fast restart time when the CPU2 clock wakes up from system LP-Stop1, LP-Stop2, LPLV-Stop1 or LPLV-Stop2, ck_icn_hs_mcu and ck_icn_ls_mcu clocks may bypass the PWRLP_TEMPO delay. This is only possible when the CPU2 power consumption is low enough to be supported by the PMIC LP mode. This function is controlled by CPU2TMPSKP in RCC_PWRLPDLYCR.

Note: In this case, the PMIC still transitions from LP to HP mode.

Refer to *Clock restore sequence* for additional informations.

Warning: **When CPU2TMPSKP is set to 1, the application must insure that the CPU2 did not allocate peripherals located into the AXI subsystem (AHB5, APB4 or APBDBG). If this rule is not respected, a system reset may be generated.**

Clock restore sequence

Every time the system exits one of the Stop modes (pwrwake_sys), the RCC reenables the requested oscillators and PLLs in order to restore the subsystem clocks. These clock settings are defined by the registers listed in *Table 124*, that cannot be modified during the clock restore sequence.

If the system exits from one of the Stop modes due to a CPU1 wake-up event (pwrwake_cpu1), the RCC enables ck_pll1_ref and ck_cpu1_ext2f clocks. All CPU1 subsystem clocks are reenabled simultaneously once all internal and external PLL ready signals are received.

The same strategy applies in case of a CPU2 wake-up event (pwrwake_cpu2) except when the selected CPU2 clock (ck_icn_hs_mcu) is an oscillator clock (HSI, HSE or MSI) before the system goes to one of the Stop modes. In this case, the CPU2 clock domain is reenabled before waiting for all internal and external PLL ready signals. It is up to the application to take care of the PLL states prior using any peripherals clocked by PLLs.

Note: When leaving Stop2, LP-Stop2 or LPLV-Stop2, the PLL1 ready signal is not taken into account.

The PLL2 clock source (ck_pll2_ref) is always reenabled when the system exits from one of the Stop modes, if DDR is shared.

More precisely, the clock restore sequence when the system exits Stop, is split into the following main steps (see *Figure 98* and *Figure 99*):

1. Set CKREST = 1 in order to inform that a clock restore sequence is initiated. Registers listed on *Table 124* cannot be modified during the clock restore sequence.
2. Enable the oscillators which were enabled before going to system Stop.
3. When the HSI is ready, the RCC can trigger the PWRLP_TEMPO to allow a delay to the external PMIC to switch in HP mode, before doing power-consuming operations on V_{DDCORE}.
4. When the oscillators used by PLLs are ready, the PLL analog parts can be powered. As these parts are not powered by V_{DDCORE}, it is possible to enable the VCOs (no need to wait that PWRLP_TEMPO elapsed).
5. In case of CPU2 wake-up event, the CPU2 clock domain is reenabled without waiting for PLLs to be ready, if the CPU2 clock source is an oscillator.
6. Before enabling the PLL outputs, the RCC must wait until the PWRLP_TEMPO elapsed, and the PLLs are ready. The process generating the CPU2 clock domain may also need to wait until the PWRLP_TEMPO elapsed when CPU2TMPSKP = 0.
7. Depending on the wake-up event, only the related CPU clock domain is reenabled. The RCC clears CKREST to 0. Registers listed in *Table 124* can then be modified by the application if needed.

Figure 98. Clock restore process activated after system Stop

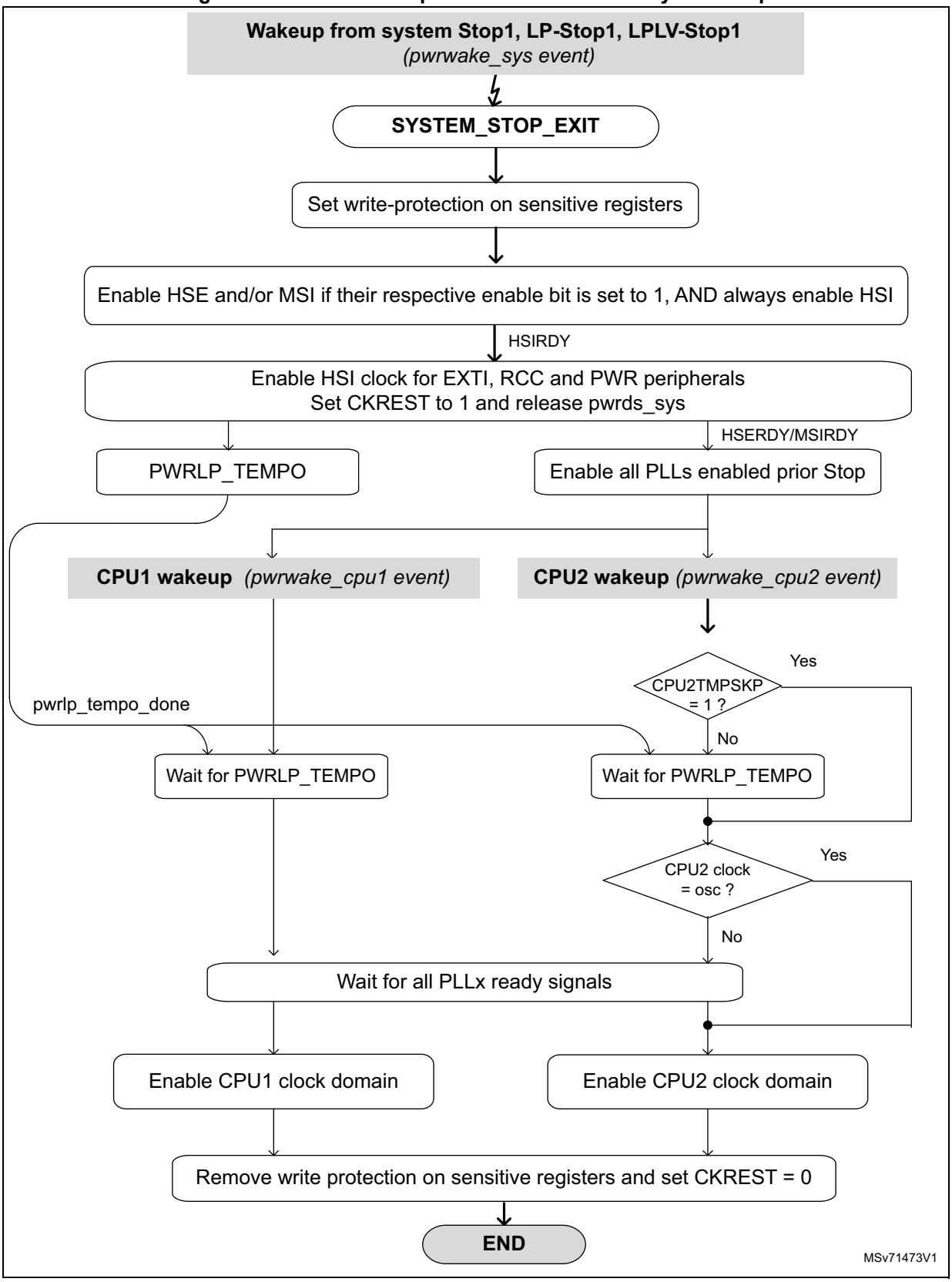
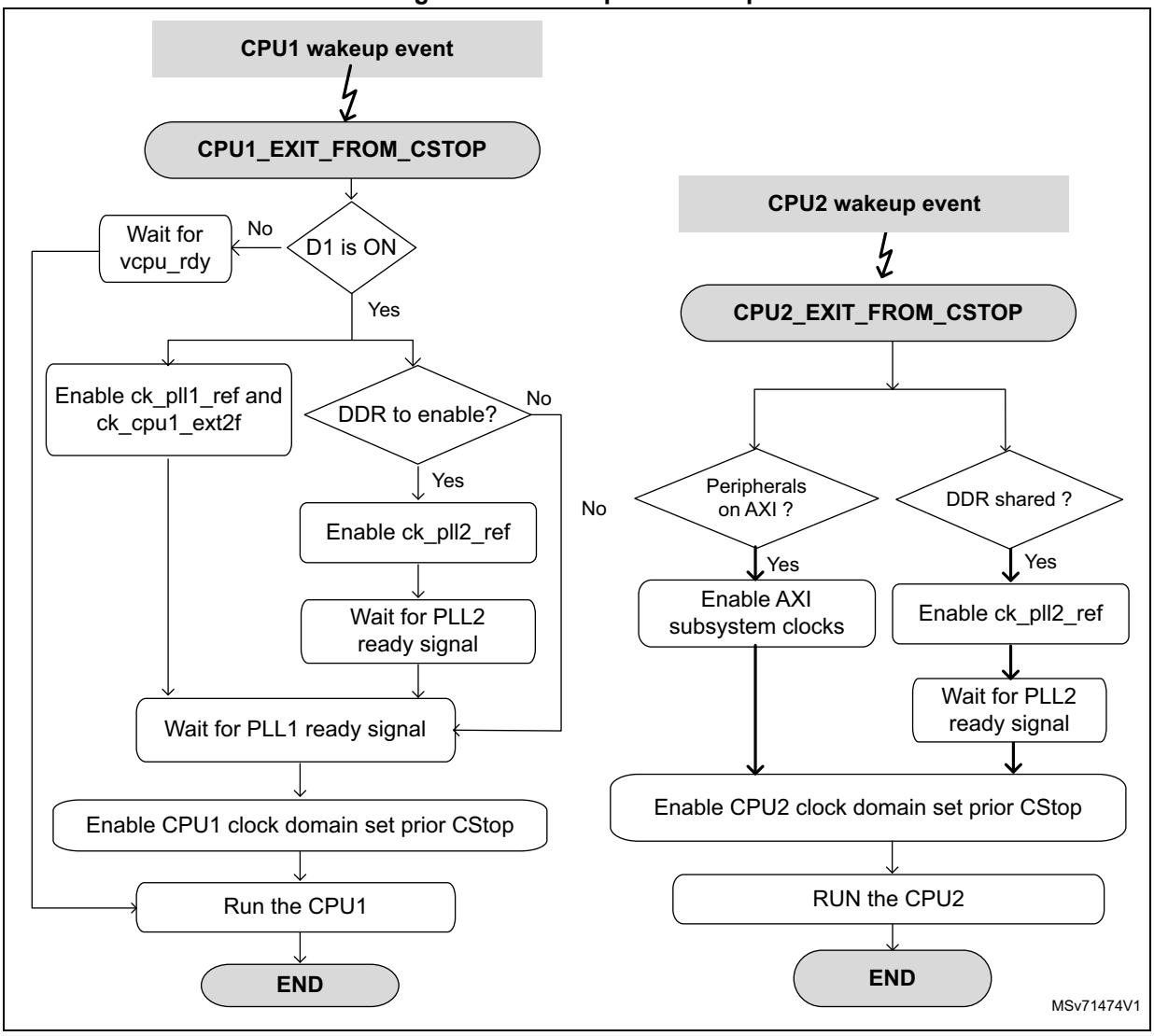


Figure 99. Wake-up from CStop



Note: When the system exits from Stop due to a wake-up event for the CPU2, if the CPU2 has a peripheral using HSE as kernel clock, the CPU2 has to check that the HSE is ready before using this peripheral.

In order to avoid any unexpected behavior during the clock restore, the RCC provides a write protection to all registers having a sensitive content for the clock restore.
The application must not attempt to write these registers when the clock restore sequence is running. If a CPUx attempts to write one of the registers listed in *Table 124*, the write operation is ignored. This situation may occur if one CPUx is already running while the other exits CStop.

Note: The RCC provides CKREST in order to inform the application if a clock restore is ongoing.

Table 124. Resources blocked during clock restore sequence

Resource	Corresponding registers write-protected during clock restore
System clock muxes	RCC_XBARxCFGR (x = 0 to 63)

Table 124. Resources blocked during clock restore sequence (continued)

Resource	Corresponding registers write-protected during clock restore
System clock dividers	RCC_PREDIVxCFGR, RCC_FINDIVxCFGR (x = 0 to 63), RCC_LSMCUDIVR
PLLx (x = 2, 4 to 8) settings	RCC_PLLxCFGR1, RCC_PLLxCFGR2, RCC_PLLxCFGR3, RCC_PLLxCFGR4, RCC_PLLxCFGR5, RCC_PLLxCFGR6, RCC_PLLxCFGR7
Reference clock selection	RCC_MUXSELCFGR
Oscillator settings	RCC_OCENSETR, RCC_OCENCLRR, RCC_HSICFGR, RCC_MSICFGR, RCC_LSICR
Miscellaneous	RCC_PWRLPDLYCR

Going to Standby mode

The system can go to Standby mode when both CPU1 and CPU2 are in CStop, without any pending wake-up event on the EXTI, and when both CPUx allow Standby mode (see *Section 19: Power control (PWR)* for details).

When the complete system goes to Standby mode, the RCC stops all the clocks and PLLs, with the exception of the LSI and LSE. The RCC also informs the PWR that CPU1 and CPU2 subsystems are in CStop via pwrds_cpu1 and pwrds_cpu2 signals (see *Figure 73*). The PWR can then request the external PMIC to switch off V_{DDCORE} and V_{DDCPU} voltages.

Only parts working on V_{DD} and V_{SW} are kept operating in Standby.

Leaving Standby mode

When the circuit exits Standby mode, the HSI is selected as default clock for the complete system, the RCC registers are reset to their initial values (except RCC_LEGBOOTCR, RCC_STBYBOOTCR, RCC_LSICR, RCC_BDCR, RCC_RDCR, RCC_HWRSTSCLRR, RCC_C1HWRSTSCLRR, and RCC_C2HWRSTSCLRR).

Note: A system reset is generated when the circuit exits Standby mode.

20.4.9 Peripheral clock distribution

Some peripherals are designed to work with two different clock domains asynchronous each other:

- a clock domain synchronous to the register and bus interface (ck_icn_p_perx clock)
- a clock domain generally synchronous to the peripheral (kernel clock ck_ker_perx)

The benefit of having peripherals supporting these two clock domains is that the user has more freedom to choose optimized clock frequency for the CPUs, bus matrix, and for the kernel part of the peripheral. The user can then change the bus frequency without reprogramming the peripherals. For example, an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly.

Table 125 displays the kernel clock that the RCC can provide to the peripherals. Each row of this table represents generally a MUX, and peripherals connected to its output. The columns starting at number 5, represent a clock source. Column 3 gives the maximum allowed frequency at each MUX output. Columns 6 up to 18 represent the possible low-power kernel clock sources (“X” means supported clock).

Table 125. Kernel clock distribution overview

Peripheral clock	Clock property (1)	Max. allowed freq. (MHz)	Bus	Flexclkgen channel number	Kernel clock source(2)												
					pll4_ck	pll5_ck	pll6_ck	pll7_ck	pll8_ck	spdifrx_symb_ck	ck_in	lse_ck	lsi_ck	msi_ker_ck	hsi_ker_ck	hse_ker_ck	timg[2:1]_ck
ck_ker_tim2, ck_ker_tim3, ck_ker_tim4, ck_ker_tim5, ck_ker_tim6, ck_ker_tim7, ck_ker_tim10, ck_ker_tim11, ck_ker_tim12, ck_ker_tim13, ck_ker_tim14	S	150	APB1	-	-	-	-	-	-	-	-	-	-	-	-	-	X
ck_ker_lptim1, ck_ker_lptim2	A	100		7	X	X	X	X	X	-	-	LP	LP	-	-	-	-
ck_ker_usart2, ck_ker_uart4	A	100		8	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_usart3, ck_ker_uart5	A	100		9	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_spi2	A	133		10	X	X	X	X	X	-	X	-	-	LP	LP	LP	-
ck_ker_spi3	A	133		11	X	X	X	X	X	-	X	-	-	LP	LP	LP	-
ck_ker_spdifrx	A	200		12	X	X	X	X	X	-	-	-	-	-	LP	-	-
ck_ker_i2c1, ck_ker_i2c2	A	200		13	X	X	X	X	X	-	-	-	-	LP	LP	-	-
ck_ker_i3c1, ck_ker_i3c2	A	100		14	X	X	X	X	X	-	-	-	-	LP	LP	-	-
Reserved	A	-		15	-	-	-	-	-	-	-	-	-	-	-	-	-
ck_ker_tim1, ck_ker_tim8, ck_ker_tim15, ck_ker_tim16, ck_ker_tim17	S	150	APB2/ AHB2	-	-	-	-	-	-	-	-	-	-	-	-	-	X
ck_ker_spi1	A	200		16	X	X	X	X	X	-	X	-	-	LP	LP	LP	-
ck_ker_spi4, ck_ker_spi5	A	133		17	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_usart1	A	133		18	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_usart6	A	100		19	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_uart7	A	100		20	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_mdf1(3)(4)	A	133		21	X	X	X	X	X	X	X	-	-	LP	LP	LP	-
ck_ker_sai1	A	133		22	X	X	X	X	X	X	X	-	-	LP	LP	LP	-
ck_ker_sai2	A	133		23	X	X	X	X	X	X	X	-	-	LP	LP	LP	-
ck_ker_sai3	A	133		24	X	X	X	X	X	X	X	-	-	LP	LP	LP	-
ck_ker_sai4	A	133		25	X	X	X	X	X	X	X	-	-	LP	LP	LP	-
ck_ker_fdcan	A	133		26	X	X	X	X	X	-	-	-	-	-	-	LP	-
ck_ker_iwdg1, ck_ker_iwdg2, ck_ker_iwdg3, ck_ker_iwdg4	A	1	APB3	-	-	-	-	-	-	-	-	-	X	-	-	-	-
ck_ker_dts (TEMP)	A	64		-	-	-	-	-	-	-	-	-	-	X	X	X	-

Table 125. Kernel clock distribution overview (continued)

Peripheral clock	Clock property (1)	Max. allowed freq. (MHz)	Bus	Flexclkgen channel number	Kernel clock source(2)												
					pll4_ck	pll5_ck	pll6_ck	pll7_ck	pll8_ck	spdifrx_symb_ck	ck_in	lse_ck	lsi_ck	msi_ker_ck	hsi_ker_ck	hse_ker_ck	timg[2:1]_ck
ck_ker_ltdc (pixel clock)	A	314	APB4	27	X	X	X	X	X	-	-	-	-	-	-	-	-
Reserved	A	-		28	-	-	-	-	-	-	-	-	-	-	-	-	-
ck_ker_dcmipp (DCMIPP+CSI)	A	333		29	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_csitxesc	A	20		30	X	X	X	X	X	-	-	-	-	-	-	X	-
ck_ker_csiphy	A	27		31	X	X	X	X	X	-	-	-	-	-	-	X	-
Reserved	A	-		32	-	-	-	-	-	-	-	-	-	-	-	-	-
ck_ker_stgen	A	64		33	X	X	X	X	X	-	-	-	-	-	X	X	-
Reserved	A	-		34	-	-	-	-	-	-	-	-	-	-	-	-	-
Reserved	A	-		35	-	-	-	-	-	-	-	-	-	-	-	-	-
ck_ker_i3c3	A	100	APB5	36	X	X	X	X	X	-	-	-	-	LP	LP	-	-
ck_ker_spi6	A	133		37	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_i2c3	A	133		38	X	X	X	X	X	-	-	-	-	LP	LP	-	-
ck_ker_lpuart1	A	133		39	X	X	X	X	X	-	-	-	-	LP	LP	LP	-
ck_ker_lptim3	A	100		40	X	X	X	X	X	-	-	LP	LP	-	-	-	-
ck_ker_lptim4	A	100		41	X	X	X	X	X	-	-	LP	LP	-	-	-	-
ck_ker_rtc (RTC/AWU(5))	A	4		-	-	-	-	-	-	-	-	X	X	-	-	X	-
ck_ker_lptim5	A	100		42	X	X	X	X	X	-	-	LP	LP	-	-	-	-
ck_ker_tsdbg	A	64	APBDBG	43	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_tpiu	A	200		44	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_sys_atb, ck_icn_m_etr	A	400	ATB	45	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_adc1	A	133	AHB2	46	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_adc2	A	133		47	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_ospi1 (OCTOSPI1)	A	266		48	X	X	X	X	X	-	-	-	-	-	-	-	-
Reserved	A	-		49	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 125. Kernel clock distribution overview (continued)

Peripheral clock	Clock property (1)	Max. allowed freq. (MHz)	Bus	Flexclkgen channel number	Kernel clock source(2)												
					pll4_ck	pll5_ck	pll6_ck	pll7_ck	pll8_ck	spdifrx_symb_ck	ck_in	lse_ck	lsi_ck	msi_ker_ck	hsi_ker_ck	hse_ker_ck	timg[2:1]_ck
ck_ker_fmc	A	200	AXI,AHB5	50	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_sdmmc1	A	200		51	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_sdmmc2	A	200		52	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_sdmmc3	A	200		53	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_eth1	A	125		54	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_eth2	A	125		55	X	X	X	X	X	-	-	-	-	-	-	-	-
ck_ker_eth1ptp, ck_ker_eth2ptp	A	200		56	X	X	X	X	X	-	-	-	-	X	X	X	-
ck_ker_usb2phy1	A	24		57	X	X	X	X	X	-	-	-	-	-	-	X	-
ck_ker_usb2phy2	A	24		58	X	X	X	X	X	-	-	-	-	-	-	X	-
Reserved	A	-		59	-	-	-	-	-	-	-	-	-	-	-	-	-
Reserved	A	-	NA	60	-	-	-	-	-	-	-	-	-	-	-	-	-
ck_mco1	A	160(6)		61	X	X	X	X	X	X	X	X	X	X	X	X	-
ck_mco2	A			62	X	X	X	X	X	X	X	X	X	X	X	X	-
ck_cpu1_ext2f	A	400		63	X	X	X	X	X	-	-	-	-	X	X	X	-

1. ‘A’ means that the kernel clock is asynchronous with respect to bus interface clock. ‘S’ means that the kernel clock is synchronous with respect to bus interface clock.
2. ‘X’ means supported kernel clock source. ‘LP’ means supported kernel clock source when the CPU that has allocated the peripheral is in CStop. hsi_ck, hse_ck, and msi_ck are possible for all Flexclkgen channels.
3. MDF1 is on AHB2 bus.
4. MDF1 low power modes (LP) requires to set HSIKERON, MSIKERON or HSEKERON bit in RCC_OCENSETR register according to the selected kernel clock sources.
5. The RTC switch is in the VSW domain. When HSE is selected, the kernel clock is divided. By default, the RTC kernel clock is off. Refer to RCC_BDCR for details.
6. Depends on I/O capability.

For CPU2, the SysTick timer clock requires synchronous clock enables, SSTCLKEN and NSSTCLKEN, that are pulsed every 8 cycles of ck_icn_hs_mcu root clock.

Debug authentication is possible when system is under reset. In this case all debug components located behind AP0 (APB-AP) access port are required to be clocked. These components are belonging to the ck_dap clock domain which is only sourced with HSI clock (hsi_ck).

The following sections show a more detailed view of the kernel clock and bus interface clock distribution (see *Section 20.4.12* for details).

In order to reduce the amount of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Clock distribution for timers

The TIMs timers use kernel clocks (ck_ker_timx, ck_ker_timy) phase aligned with the bus interface clock. The frequency of the timer clocks depends on:

- the APB prescaler corresponding to the bus where the timer is connected
- TIMG[2:1]PRE bits

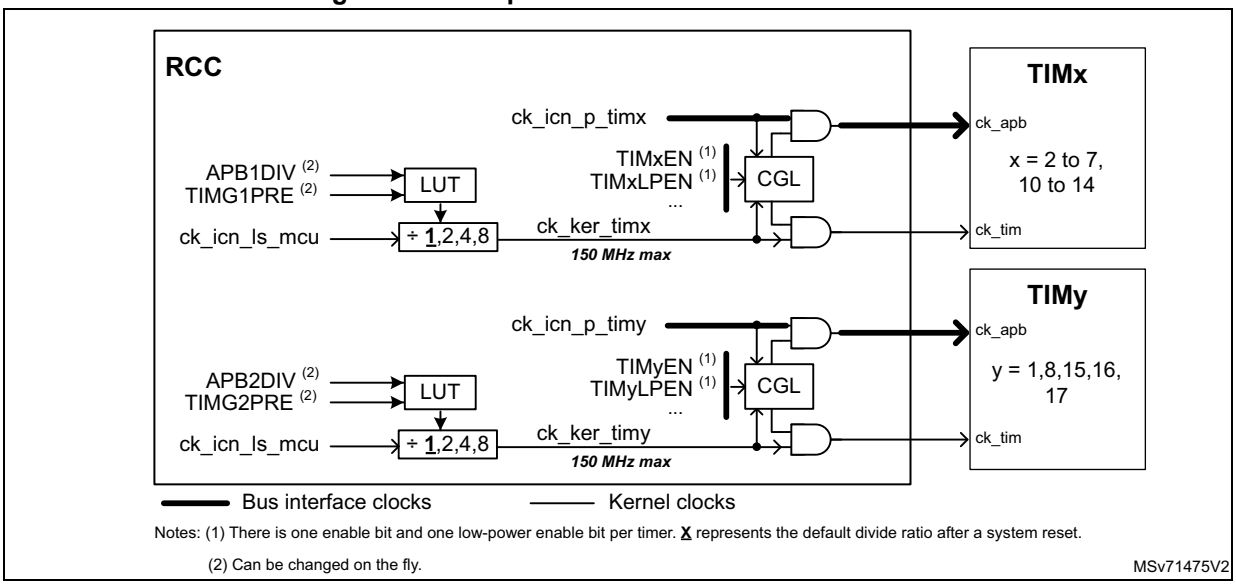
The table below shows how to select the timer clock frequency.

Table 126.Ratio between clock timers and ck_icn_apbx

APB1DIV APB2DIV(1)	TIMG1PRE TIMG2PRE(2)	$F_{ck_ker_timx}$ $F_{ck_ker_timy}$	$F_{ck_icn_apb1}$ $F_{ck_icn_apb2}$	Comments
000	x	$F_{ck_icn_ls_mcu}$	$F_{ck_icn_ls_mcu}$	The timer clock is equal to the bus clock.
001	x	$F_{ck_icn_ls_mcu}$	$F_{ck_icn_ls_mcu}$ / 2	The timer clock is two times faster than the bus clock.
010	0	$F_{ck_icn_ls_mcu}$ / 2	$F_{ck_icn_ls_mcu}$ / 4	
011	0	$F_{ck_icn_ls_mcu}$ / 4	$F_{ck_icn_ls_mcu}$ / 8	
100	0	$F_{ck_icn_ls_mcu}$ / 8	$F_{ck_icn_ls_mcu}$ / 16	
010	1	$F_{ck_icn_ls_mcu}$	$F_{ck_icn_ls_mcu}$ / 4	The timer clock is four times faster than the bus clock.
011	1	$F_{ck_icn_ls_mcu}$ / 2	$F_{ck_icn_ls_mcu}$ / 8	
100	1	$F_{ck_icn_ls_mcu}$ / 4	$F_{ck_icn_ls_mcu}$ / 16	

1. APB1DIV and APB2DIV are in RCC_APB1DIVR and RCC_APB2DIVR.
2. TIMG1PRE and TIMG2PRE are in RCC_TIMG1PRER and RCC_TIMG2PRER.

Figure 100. Peripheral clock distribution for TIMs



Notes: (1) There is one enable bit and one low-power enable bit per timer. X represents the default divide ratio after a system reset.

(2) Can be changed on the fly.

MSv71475V2

Refer to *Section 20.4.12* for details on the behavior of the clock gating logic (CGL).

Clock distribution for ADCs

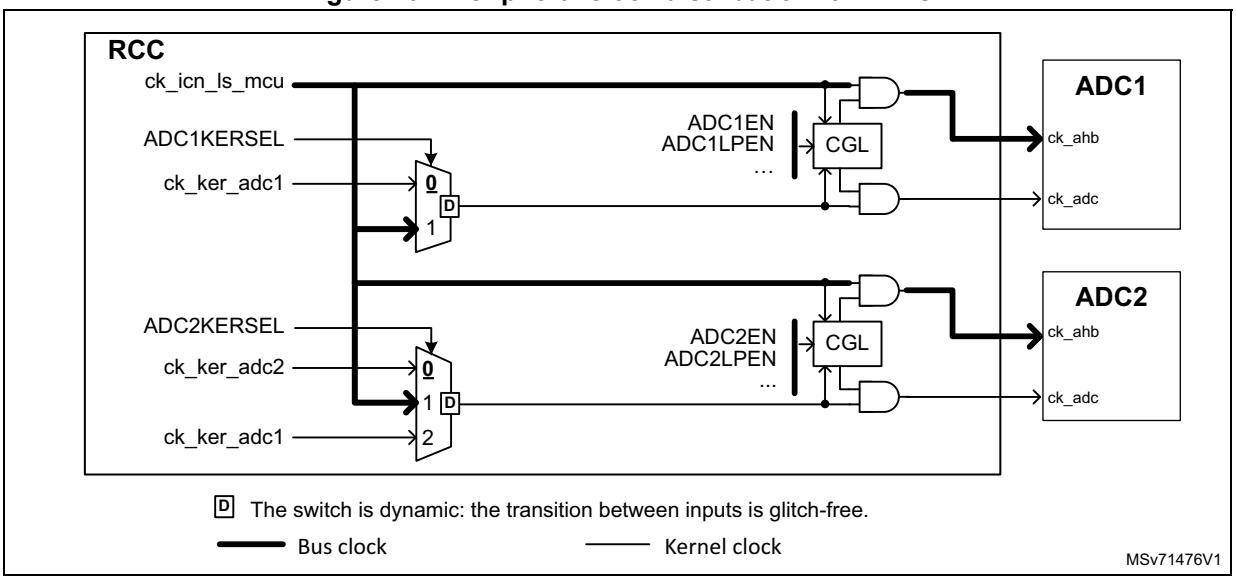
The ADC kernel clocks can be selected among the following source clocks: ck_ker_adc1, ck_ker_adc2 or ck_icn_ls_mcu.

Depending on the accuracy of measured values at ADC level, their kernel clocks can be selected to be either synchronous or asynchronous, according to ADC1KERSEL and ADC2KERSEL[1:0] in RCC_ADC1CFGR and RCC_ADC2CFGR.The table below shows how to select the kernel clocks source of the 2 ADCs.

Table 127. ADC kernel clock source selection

ADC1KERSEL	ADC2KERSEL[1:0]	ADC1 ck_adc source	ADC2 ck_adc source	Comments
~~0~~	0x0	ck_ker_adc1	ck_ker_adc2	Asynchronous mode
	0x1		ck_icn_ls_mcu	
	0x2		ck_ker_adc1	Synchronous mode
	0x3		NA	-
1	0x0	ck_icn_ls_mcu	ck_ker_adc2	Asynchronous mode
	0x1		ck_icn_ls_mcu	Synchronous mode
	0x2		ck_ker_adc1	Asynchronous mode
	0x3		NA	-

Figure 101. Peripheral clock distribution for ADCs



Clock distribution for the DDR

Figure 102 shows a simplified view of the clock distribution for the different blocks inside the DDR subsystem (DDRCTRL, DDRPHYC, DDRPERFM, RISAF4, MCE, PLL2).

The RCC needs to provide the following clocks:

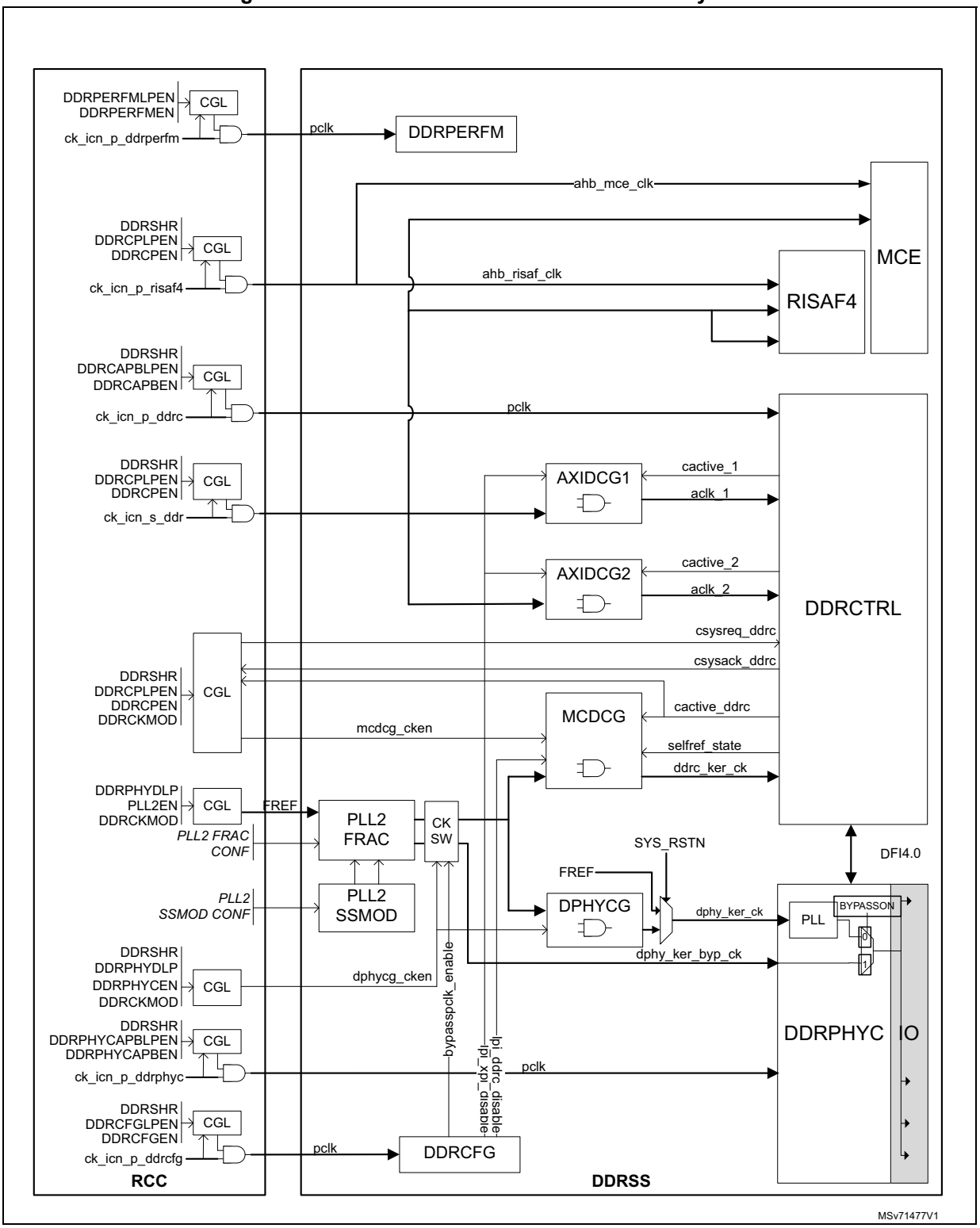
- APB and AHB clocks for the register interface (ck_icn_p_ddrc, ck_icn_p_ddrphyc, ck_icn_p_ddrperfm, ck_icn_p_risaf4 and ck_icn_p_ddrcfg)
- AXI clock for the three AXI ports of DDRCTRL, RISAF4 and MCE (ck_icn_s_ddr)

- reference clock for the PLL2

The RCC needs to provide also the following control bits:

- MCDCG control bit (mcdcg_cken). Refer to *Table 134* and *Table 135* for details.
- DPHYCG control bit (dphycg_cken). Refer to *Table 136* and *Table 137* for details.
- PLL2 configuration bits. Refer to *Section 20.8.95* to *Section 20.8.101* for details.

Figure 102. Clock distribution for the DDR subsystem



Control of APB and AHB clocks

The control of the APB/AHB clock gating is performed via the following bits:

- DDRSHR, DDRCAPBEN, and DDRCAPBLPEN control the gating of the APB clock of the DDRCTRL interface (ck_icn_p_ddrc).
- DDRSHR, DDRPHYCAPBEN, and DDRPHYCAPBLPEN control the gating of the APB clock of the DDRPHYC interface (ck_icn_p_ddrphyc).
- DDRSHR, DDRCFGEN, and DDRCFGLPEN control the gating of the APB clock of the DDRCFG interface (ck_icn_p_ddrcfg).
- DDRSHR, DDRCPEN, and DDRCPLPEN control the gating of the AHB clock of the RISAF4/MCE interface (ck_icn_p_risaf4).
- DDRPERFMEN and DDRPERFMLPEN control the gating of the APB clock of the DDRPERFM interface (ck_icn_p_ddrperfm).

The dedicated RCC control bits for the DDR (DDRSHR, DDRCKMOD and DDRPHYDLP) are located in RCC_DDRITFCFGR.

Table 128. Gating control of the APB clocks with ck_icn_p_ddrc and ck_icn_p_ddrphyc

DDRxEN(1)	DDRxLPEN(1)	DDRSHR(2)	CPU1 state	System mode	ck_icn_p_apbx clock	Comments
0	X	X	Any	Run	OFF	ck_icn_p_apbx is gated because DDRxEN = 0.
1	X	0	CRun	Run	ON	ck_icn_p_apbx is forced ON because DDRxEN = 1, DDRSHR = 0, and CPU1 is in CRun.
1	0	0	CSleep	Run	OFF	ck_icn_p_apbx is gated because DDRxLPEN = 0, and CPU1 is in CSleep.
	1				ON	ck_icn_p_apbx is forced ON because DDRxLPEN = 1, and CPU1 is in CSleep.
1	X	0	CStop	Any	OFF	ck_icn_p_apbx is gated because CPU1 is in CStop.
1	X	1	Any	Run	ON	ck_icn_p_apbx is forced ON because DDRSHR = 1 and system is in Run.
1	X	1	Any	Stop or Standby	OFF	ck_icn_p_apbx is gated.

1. DDRxEN and DDRxLPEN are respectively the enable and low-power enable bits of any DDR APB port.
2. DDRSHR = 0 means that the DDR is only used by CPU1. Otherwise, the DDR is shared by both CPU1 and CPU2.

Control of AXI clocks

AXIDCG[1:2] handle the gating of the AXI clocks (aclk_1 and aclk_2) provided to the DDRCTRL AXI ports. ck_icn_s_ddr clock is the source clocks of the AXI clocks.

Table 129 details the gating control of ck_icn_s_ddr clock inside the RCC.
When LPI_XPI_DISABLE is cleared to 0 (see *Section 12: DDR subsystem architecture (DDRSS)* for details), the AXI clocks (aclk_1 and aclk_2) are gated according to the cactive_[1:2] signals provided by the DDRCTRL.

Table 129. Gating control of ck_icn_s_ddr clock

DDRCPEN	DDRCPLPEN	DDRSHR	CPU1 state	System mode	ck_icn_s_ddr clock	Comments
0	X	X	Any	Run	OFF	ck_icn_s_ddr is gated because DDRCPEN = 0’
1	X	0	CRun	Run	ON(1)	ck_icn_s_ddr is forced ON because DDRCPEN = 1, DDRSHR = 0, and CPU1 is in CRun.
1	0	0	CSleep	Run	OFF	ck_icn_s_ddr is gated because DDRCPLPEN = 0 and CPU1 is in CSleep.
	1				ON(1)	ck_icn_s_ddr is forced ON because DDRCPLPEN = 1
1	X	0	CStop	Any	OFF	ck_icn_s_ddr is gated.
1	X	1	Any	Run	ON(1)	ck_icn_s_ddr is forced ON because DDRSHR = 1 and system is in Run.
1	X	1	Any	Stop or Standby	OFF	ck_icn_s_ddr is gated.

1. When LPI_XPI_DISABLE = 0 locally in the DDR subsystem, ck_icn_s_ddr is gated dynamically according to **cactive_x**. This allows ASR and HSR modes to be supported.

Control of DDRCTRL and DDRPHYC kernel clocks

MCDCG and DPHYCG handle respectively the clock gating of ddrc_ker_ck, dphy_ker_ck, and dphy_ker_byp_ck. The source clocks come from the PLL2. The control bits come from the RCC, and are gated according to the way the DDRCTRL and the RCC are programmed.

Note: As some part of DDRPHYC is reset synchronously to dphy_ker_ck clock (DfiClk), PLL2 reference clock (FREF) is provided to DDRPHYC after any system reset to avoid extra power consumption.

The table below details how ddrc_ker_ck, dphy_ker_ck, and dphy_ker_byp_ck clocks are handled according to DDRCKMOD value.

Table 130. DDRCKMOD control description

DDRCKMOD	MCDCG mode	DPHYDCG mode	Comment
00	Normal	Normal	**Software self-refresh (SSR)**(1): The gating of the ddrc_ker_ck clock depends on DDRCPEN, DDRCPLPEN, DDRSHR bits, and CPU1 or system mode. The gating of dphy_ker_ck and dphy_ker_byp_ck clocks depends on DDRPHYCEN, DDRPHYDLP bits, and CPU1 or system mode.
01	Auto	Normal	**Automatic self-refresh (ASR)**(2): The clock ddrc_ker_ck is gated automatically according to cactive_ddrc signal in MCDCG. In this mode, DDRPHYCEN must be set to 1. DDRCPEN is still used to disable/enable the DDR interface
10	DDRC_LP	Normal	**Hardware self-refresh mode (HSR)**(2): The gating of the ddrc_ker_ck clock is controlled by the RCC via the DDRCTRL low-power interface. In this mode, DDRPHYCEN must be set to 1. DDRCPEN is still used to disable/enable the DDR interface.

1. This mode must be selected during DDRCTRL and DDRPHYC initialization phase, and if the application is using the software self-refresh.
2. This mode can be used when the application wants to maintain a clock to the DDRPHY.

DDRCTRL low-power interface

The DDRCTRL provides a low-power interface which can be used to control the entry/exit of the DDRCTRL and DDRPHYC into LP1 low-power mode (see *Section 12* for details). This interface controls also the gating of ddrc_ker_ck clock. This low-power interface is compliant to AMBA3 AXI LP protocol specification. In order to simplify the drawings, this document only shows the most significant states of the DDRCTRL low-power interface (see *Section 12* for details).

The table below gives details on the DDRCTRL low-power states. The DDRCTRL low-power interface can be used in HSR mode in order to control:

- the entry/exit of the external DDR device into low-power (self-refresh)
- the generation of the DDRCTRL kernel clock

Table 131. DDRCTRL low-power states description

DDRCTRL low-power states	Comment
RUN	Normal clocked operation: MCDCG and DPHYCG provide the kernel clock(s) to DDRCTRL and DDRPHYC. csysreq_ddrc, csysack_ddrc and cactive_ddrc are all set to 1 by hardware.
LP1	Low-power operation: MCDCG does not provide the kernel clock to the DDRCTRL while the DDRPHYC is kept clocked. csysreq_ddrc, csysack_ddrc and cactive_ddrc are all set to 0. The PLL2 is kept running.
LP1_Rreq	Low-power entry initiated by the RCC: The RCC requests the DDRCTRL to enter into low-power by setting csysreq_ddrc to 0. The DDRCTRL may accept or deny the request. If a request is denied, the interface comes back to RUN state, and the RCC retries a new low-power request entry. When the low-power request is accepted by the DDRCTRL (csysack_ddrc and cactive_ddrc = 0), the RCC sets MCDCG_CKEN bit to disable the DDRCTRL kernel clock. The state switches to LP1.
LP1_Rexit	Low-power exit initiated by the RCC: The RCC sets csysreq_ddrc to 1, and waits for csysack_ddrc and cactive_ddrc set to 1 before going to RUN state.
LP1_Dexit	Low-power exit initiated by the DDRCTRL: The DDRCTRL requests the kernel clock by setting cactive_ddrc to 1. The RCC sets csysreq_ddrc to 1, and waits for csysack_ddrc set to 1 before going to RUN.

Self-refresh mode control

Switching the external DDR device into self-refresh is important for energy saving. When the external DDR device is in self-refresh, the application can switch off DDRCTRL and DDRPHYC clocks without loosing the external DDR content. When the external DDR device is in self-refresh, the application can no longer access to the content of the external DDR device.

Exiting a DDR device from self-refresh can take several microseconds. The application has to take into account this latency that is due to the external DDR device, but also to the way the PLL located into the DDRPHYC is handled.

The RCC offers three ways to control the DDRCTRL kernel clock:

- software self-refresh mode (SSR)
- automatic self-refresh mode (ASR)
- hardware self-refresh mode (HSR)

There is only one way to control the DDRPHYC kernel clock:

- software self-refresh mode (SSR)

The RCC behavior is different according to the self-refresh mode selected by the application. The application has to program DDRCKMOD in accordance with DDRCTRL and DDRPHYC configuration.

Software self-refresh mode (SSR)

In this mode, the application has to program directly the DDRCTRL in order to switch the external DDR device immediately into self-refresh. Typically, when the CPU1 wants to go to CSleep or CStop, and the DDR is not shared with the CPU2 (DDRSHR = 0), the CPU1 must insure that the DDR resource is not longer needed, and program the DDRCTRL in order to switch it to self-refresh.

The piece of code performing the entry/exit of the DDRCTRL to self-refresh must be located into the SYSRAM because some software routines need to run when the DDR device is in self-refresh.

Automatic self-refresh mode (ASR)

In this mode, the software does not need to program the DDRCTRL every time the DDRCTRL must be set into self-refresh. The DDRCTRL goes automatically into self-refresh mode when the DDRCTRL is inactive for an amount of time defined into the DDRCTRL.

Even when the system is in Run mode, the DDRCTRL may go to self-refresh if there is no activity on its AXI ports. The automatic self-refresh mode allows the entry into self-refresh whatever CPUx mode.

When the CPU1 goes to CSleep, and the DDR is not shared with the CPU2, if the clock of the DDRCTRL needs to be gated, the MCDCG delays this clock gating until the DDRCTRL switches into self-refresh mode.

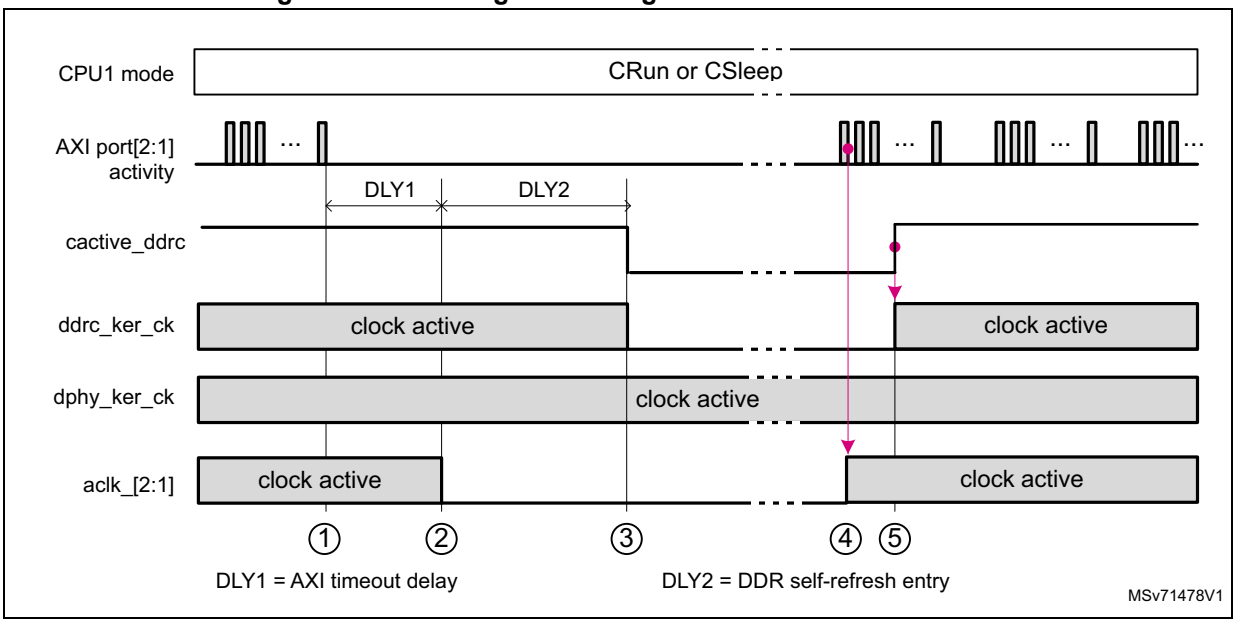
The ASR mode must be used only if the DDRCTRL has been programmed in automatic self-refresh mode, otherwise the behavior of the DDR interface is not guaranteed.
This mode can only be used when system is in Run mode.

The ASR mode impacts the gating of the DDRCTRL kernel clock (ddrc_ker_ck). dphy_ker_ck must be kept running in this mode.

When the ASR mode is used, DDRCPEN must be set to 1.

The figure below shows a simplified timing for the entry and exit from self-refresh. In this example, DDRCPEN = DDRCPLPEN = 1, DDRPHYCEN = 1, LPI_XPI_DISABLE = 0, and DDRSHR = 0.

Figure 103. Entering and exiting self-refresh in ASR mode



1. The activity on AXI port x is stopped, due for example to the fact that the CPU1 goes to CSleep, but it can be also the case between two activity bursts.
2. When no activity is detected in the AXI port x for at least UMCTL2_AXI_LOWPWR_NOPXCNT * aclk_x cycles (DLY1), the AXI clock (aclk_x) is gated if LPI_XPI_DISABLE is cleared to 0 (see *Section 12: DDR subsystem architecture (DDRSS)* for details).
3. When the AXI interfaces are gated, if they do not switch back to active after DLY2 delay, ddrc_ker_ck is gated. The CPU1 can be in CRun or CSleep.
4. If a transaction is performed in one of the two AXI ports, the AXI clock (aclk_x) is reenabled. The AXI transaction can be caused by any master using DDR resources.
5. Few AXI clock cycles later, the DDRCTRL sets cactive_ddrc to 1 in order to request the kernel clock. ddrc_ker_ck is reenabled. The external DDR is then ready to process the pending transactions.

Hardware self-refresh mode (HSR)

This mode uses the DDRCTRL low-power interface (DDRCTRL-LP) in order to request to the DDR to enter or exit LP1 low-power mode (see *Section 12* for details). The HSR mode can only be used when system is in Run mode.

The DDRCTRL-LP allows the entry of the DDR interface into self-refresh in the following cases:

- when the interface is disabled via DDRCPEN bit
- when CPU1 transitions from CRun to CSleep, and the DDR is not shared with CPU2 (DDRSHR = 0)
- when CPU1 transitions from CRun to CStop, the DDR is not shared with CPU2 (DDRSHR = 0), and the CPU2 is not in CStop.

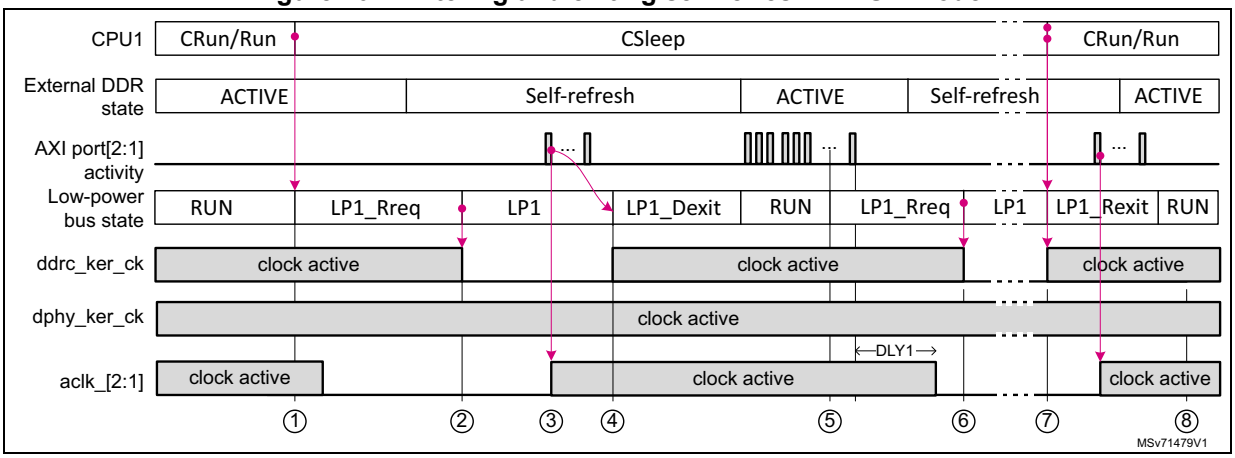
The DDRCTRL-LP allows the exit of the DDR interface from self-refresh in the following cases:

- when the interface is enabled via DDRCPEN bit
- when the CPU1 exits CSleep or CStop, and the DDR is not shared with CPU2 (DDRSHR = 0)
- when the DDRCTRL requests to exit self-refresh. This may happen if another master accesses one of the AXI ports when CPU1 is in CSleep and DDR is not shared, or when the system is in Run and DDR is shared.

This HSR mode may be less efficient in term of power saving than the ASR mode but it prevents the DDRCTRL to go accidentally to self-refresh when CPU1 is in CRun, increasing the system performances.

Figure 104 shows simplified timing for the self-refresh entry and exit. This is typically what happens when the CPU1 goes to CSleep while some masters are still enabled.
This example is based on *Table 131* configuration with DDRCPEN = DDRCPLPEN = 1, DDRPHYCEN = 1, DDRPHYDLP = 0, LPI_XPI_DISABLE = 0, and DDRSHR = 0.

Figure 104. Entering and exiting self-refresh in HSR mode



1. When the CPU1 goes to CSleep, the RCC requests to the DDRCTRL to go to self-refresh, using the DDRCTRL low-power interface (LP1_Rreq). The AXI clock is disabled if no transactions occur on AXI ports for a programmed DLY1 delay and LPI_XPI_DISABLE = 0 (see *Section 12* for details).
2. When all conditions in the DDRCTRL are met (no transaction pending, DDR in self-refresh), the DDRCTRL acknowledges the request. If the DDRCTRL denies the request, the RCC has to generate a new request until it is accepted by the DDRCTRL, or if the CPU1 exits from CSleep.

   When the DDRCTRL acknowledges the low-power request, the RCC disables ddrc_ker_ck. The DDRCTRL low-power bus is then in LP1 state.
3. If a transaction is performed in one of the two AXI ports, the AXI clock (aclk_x) is reenabled. The AXI transaction can be caused by any master using DDR resources.
4. Few AXI clock cycles later, the DDRCTRL requests to the RCC to re-enable the kernel clock, using the DDRCTRL low-power interface (LP1_Dexit). As soon as the RCC receives the request, ddrc_ker_ck is reenabled. The external DDR is then ready to process the pending transactions.
5. As the CPU1 is still in CSleep, the RCC requests to the DDRCTRL to go to self-refresh, using the DDRCTRL low-power interface (LP1_Rreq). The DDRCTRL may deny the

request several times if all conditions to enter in self-refresh are not met. The RCC has to renew its request regularly until it is accepted by the DDRCTRL. The RCC must stop performing this request if the CPU1 goes to CRun meanwhile.

6. When all conditions in the DDRCTRL are met (no transaction pending, DDR in self-refresh), the DDRCTRL acknowledges the request. When the DDRCTRL acknowledges the low-power request, the RCC disables ddrc_ker_ck. The DDRCTRL low-power interface is then in LP1 state.

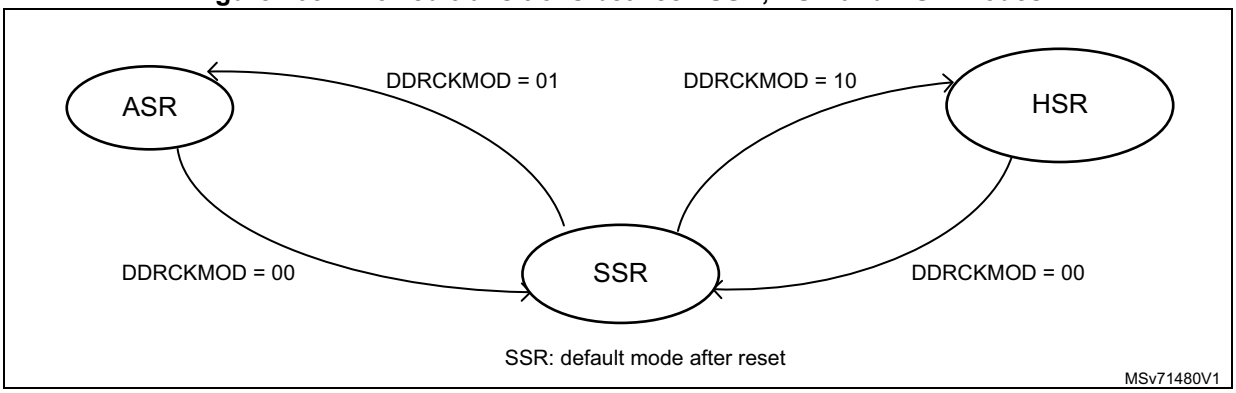
Note: The sequence described between step 4 to step 7 may be repeated several times until the CPU1 is in CSleep.

7. When the CPU1 exits from CSleep, the RCC reenables ddrc_ker_ck, and requests to the DDRCTRL to exit the DDR from self-refresh, using the DDRCTRL low-power bus (LP1_Rreq).
8. The DDRCTRL acknowledges the request by setting the DDRCTRL low-power interface into RUN state. The external DDR is then ready to process the pending transactions.

Transitions between SSR, ASR and HSR

The application can change DDRCKMOD value without disabling the DDR interface. The figure below shows the allowed transition between SSR, ASR and HSR modes.

Figure 105. Allowed transitions between SSR, ASR and HSR modes



The following direct transitions are not allowed:

- From ASR to HSR: the application needs first to set DDRCKMOD[1:0] to 0x0 (SSR), and then to set DDRCKMOD[1:0] to 0x2 (HSR mode).
- From HSR to ASR: the application needs first to set DDRCKMOD[1:0] to 0x0 (SSR), and then to set DDRCKMOD[1:0] to 0x1 (ASR mode).

Control of PLL2 reference clock

In all DDR self-refresh modes (SSR, ASR or HSR) described before, the PLL2 always runs. If the application wants to set more power savings (gated dphy_ker_ck and dphy_ker_byp_ck, and powered down PLL2), it has to set the DDRPHY in deep low-power mode (LP2 or LP3 low-power mode). This is only possible in SSR mode.

DDRPHYDLP in RCC_DDRITFCFGR is used to notify deep low-power mode for the RCC.

Recommended configurations

DDRCPEN and DDRPHYCEN bits are used to enable/disable the DDR interface.

Table 132 and *Table 133* show the recommended RCC configurations. The application has to select the configuration adapted to its DDR configuration.

Note: In configurations #4 and #8, the PLL2 is powered down by hardware.

Table 132. Recommended configurations when DDR is not shared (DDRSHR = 0)

Conf #	DDRCKMOD	DDRCPEN	DDRCPLPEN	DDRPHYCEN	DDRPHYDLP	CPU1 CRun			CPU1 CSleep			CPU1 CStop			Comments
						AXI clocks (1)	DDRCTRL clock(2)	DDRPHYC clock	AXI clocks (1)	DDRCTRL clock(2)	DDRPHYC clock	AXI clocks (1)	DDRCTRL clock	DDRPHYC clock	
1	Any	0	x	0	x	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Interface disabled
2	SSR	1	1	1	0	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	OFF	OFF	**ON**	Gating of DDRCTRL and DDRPHYC clocks depends on CPU1 state, and DDRPHYDLP. The application software must switch the DDR interface into a low-power mode, before gating clocks.
3		1	0	1	0	**ON**	**ON**	**ON**	OFF	OFF	**ON**	OFF	OFF	**ON**	
4		1	0	1	1	**ON**	**ON**	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	
5	ASR	1	1	1	0	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	OFF	OFF	**ON**	Only the DDRCTRL kernel clock is controlled according to automatic mode
6	HSR	1	0	1	0	**ON**	**ON**	**ON**	OFF	**DLP**	**ON**	OFF	OFF	**ON**	The DDRCTRL kernel clock is controlled according to DDRCTRL-LP interface. The DDRPHYC kernel clock must be maintained active by setting DDRPHYCEN = 1.

1. When LPI_XPI_DISABLE = 0, the AXI clock is gated automatically according to **c**active_[2:1] (DCG), except when DDRCPEN = 0.
2. Dynamic clock gating is supported according to cactive_ddrc and selfref_state.

Table 133.Recommended configurations when DDR is shared (DDRSHR = 1)

Conf #	DDRCKMOD	DDRCPEN	DDRCPLPEN	DDRPHYCEN	DDRPHYDLP	System Run			System Stop1/2			Comments
						AXI clocks (1)	DDRCTRL clock(2)	DDRPHYC clock	AXI clocks (1)	DDRCTRL clock(2)	DDRPHYC clock	
7	SSR	1	x	1	0	**ON**	**ON**	**ON**	NA			Gating of DDRCTRL and DDRPHYC clocks depends on system mode and DDRPHYDLP. The application software must switch the DDR interface into a low-power mode, before gating the clocks.
8		1	x	1	1	**ON**	**ON**	**ON**	OFF	OFF	OFF	
9	ASR	1	x	1	0	**ON**	**ON**	**ON**	NA			Only DDRCTRL kernel clock is controlled according to the automatic mode
10	HSR	1	x	1	0	**ON**	**DLP**	**ON**	NA			The DDRCTRL kernel clock is controlled according to DDRCTRL-LP interface. LP1_Rreq is only possible when DDRCPEN = 0 in this case. The DDRPHYC kernel clock must be maintained active by setting DDRPHYCEN = 1.

1. When LPI_XPI_DISABLE = 0, the AXI clock is gated automatically according to cactive_[2:1] (DCG), except when DDRCPEN = 0.
2. Dynamic clock gating is supported according to cactive_ddrc and selfref_state. Only valid in SSR and ASR modes.

MCDCG control interface

The tables below detail the mcdcg_cken control value in all DDR use cases.

Table 134. MCDCG control bit when DDR is not shared (DDRSHR = 0)

DDRCKMOD	DDRCPEN	DDRCPLPEN	CPU1 state	mcdcg_cken	Comments
SSR	0	x	Any	0	DDR not used
	1	x	CRun	1	Application put the DDR in self-refresh.
	1	0	CSleep	0	
	1	1		1	
	1	x	CStop	0	
ASR(1)	0	x	Any	0	DDR not used
	1	x		1	Gated locally when the DDR is in self-refresh

Table 134. MCDCG control bit when DDR is not shared (DDRSHR = 0) (continued)

DDRCKMOD	DDRCPEN	DDRCPLPEN	CPU1 state	mcdcg_cken	Comments
HSR (1)	0	x	Any	0	DDR not used
	1	x	CRun	1	No gating
	1	0	CSleep	0	Gated locally once LP1 state is reached after LP1_Rreq
	1	1		1	No gating
	1	x	CStop	0	Gated locally once LP1 state is reached after LP1_Rreq

1. Only valid when system is in Run mode.

Table 135. MCDCG control bit when DDR is shared (DDRSHR = 1)

DDRCKMOD	DDRCPEN	DDRCPLPEN	System mode	mcdcg_cken	Comments
SSR	0	x	Any	0	DDR not used
	1	x	Run	1	Application has put the DDR in self-refresh.
	1	x	Stop	0	
ASR	0	x	Run	0	DDR not used
	1	x		1	Gated locally when the DDR is in self-refresh
HSR	0	x	Run	0	Gated (0) once LP1 state is reached after LP1_Rreq
	1	x		1	No gating

DPHYCG control interface

The tables below detail the dphycg_cken control value in all DDR use cases.

Table 136. DPHYCG control bit when DDR is not shared (DDRSHR = 0)

DDRCKMOD	DDRPHYCEN	DDRPHYDLP	CPU1 state	dphycg_cken	Comments
SSR	0	x	Any	0	No gating
	1	x	CRun	1	No gating
	1	0	CSleep	1	No gating
	1	1		0	Application manages self-refresh and PLL2 power down.
	1	0	CStop	1	No gating
	1	1		0	PLL2 is powered down by hardware.

Table 136. DPHYCG control bit when DDR is not shared (DDRSHR = 0) (continued)

DDRCKMOD	DDRPHYCEN	DDRPHYDLP	CPU1 state	dphycg_cken	Comments
ASR(1)	0	x	Any	0	No gating
	1	x		1	No gating
HSR (1)	0	x	Any	0	No gating
	1	x		1	No gating

1. Only valid when system is in Run mode.

Table 137. DPHYCG control bit when DDR is shared (DDRSHR = 1)

DDRCKMOD	DDRPHYCEN	DDRPHYDLP	System mode	dphycg_cken	Comments
SSR	0	x	Any	0	No gating
	1	0(1)	Run	1	No gating
	1	0	Stop	1	No gating
	1	1		0	Application manages self-refresh. PLL2 is powered down by hardware.
ASR	0	x	Run	0	No gating
	1	x		1	No gating
HSR	0	0	Run	0	No gating
	1	0		1	No gating

1. DDRPHYDLP = 1 is not applicable in Run mode.

20.4.10 General clock concept overview

The RCC handles the distribution of the CPU, bus interface, and peripheral clocks for the system, according to the operating mode of each CPUx (see *Section 19: Power control (PWR)* for details on operating modes and *Section 20.4.1* for clock definitions).

For most of the peripherals, it is possible for the application to control the activation/deactivation of their kernel and bus interface clocks.

Prior to use a peripheral, the CPU1 or CPU2 has to enable it (setting its PERxEN to 1), and to define if this peripheral remains active in CSleep (setting its PERxLPEN to 1).
This operation is called ‘allocate’ a peripheral to a CPUx (see *Section 20.4.11* for details).
The peripheral allocation is used by the RCC to control automatically the clock gating according to CPU states.

For example if the CPU1 enables a peripheral and decides that the kernel clock of this peripheral must be gated when it goes to CSleep, the CPU2 state has no effect on the clock gating of this peripheral, but the clock gating obeys to CPU1 state. This is true if the peripheral is not shared (see *Section 20.4.11* for details). Allocating a peripheral based on

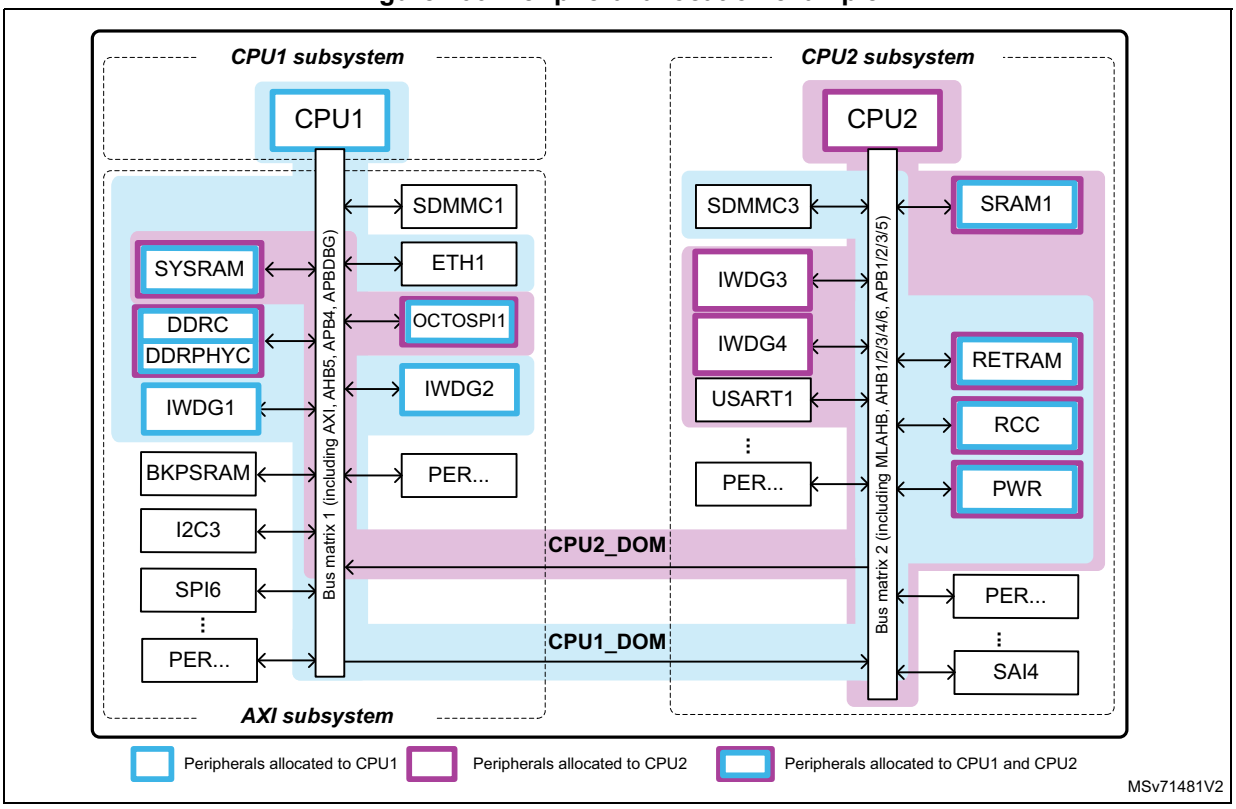
its RIF configuration guarantees to a CPUx that this peripheral remains accessible, whatever the other CPUx mode.

Figure 106 gives an example of peripheral allocation:

- CPU1 enabled ETH1, SDMMC3 and RETRAM. The group composed by the CPU1, bus matrix 1, bus matrix 2, and peripherals allocated via RCC_PERxCFGR form the CPU1 peripheral domain (CPU1_DOM).
- CPU2 enabled SYSRAM, OCTOSPI1, SRAM1 and USART1. The group composed by the CPU2, bus matrix 1, bus matrix 2, and peripherals allocated via RCC_PERxCFGR form the CPU2 peripheral domain (CPU2_DOM).

Note: IWDG1 and IWDG2 are expected to be allocated only to CPU1. IWDG3 and IWDG4 are expected to be allocated only to CPU2. PWR and RCC are common resources and allocated to both CPU1 and CPU2. Memories are allocated to both CPUx (see Figure 106 and Handling of memory devices for more details).

Figure 106. Peripheral allocation example



Note: CPU1_DOM and CPU2_DOM can be spread over the two domains.

When the CPU1 goes to CStop, the RCC disables automatically the bus interface and kernel clocks of all peripherals of CPU1_DOM, as well as CPU1 clock. If CPU2 is still in CRun, the bus matrix 1 is still clocked as CPU2_DOM contains peripherals (SYSRAM, OCTOSPI1) physically connected to this bus matrix. The PLLs, if enabled, are not disabled by the RCC as the CPU2 still runs.

If CPU2 goes to CStop, the bus matrix 1 is no longer clocked.

Note: In this scenario, the complete system can go to Stop. If both CPUs allowed Standby mode, the system goes to Standby.

The wake-up events can exit the system from Standby or Stop.

Handling of memory devices

Both CPU1 and CPU2 can access all memory devices available in the product:

- SYSRAM
- SRAM1, OCTOSPI1
- RETRAM, BKPSRAM

By default, the RETRAM memory is allocated (enabled) to both CPUx to allow accesses when the system exits Standby. All others memories are not allocated by default.

According to the RCC RIF resource configuration, any internal memory can be dedicated or shared by CPUs (see *Section 20.4.11* for details).

Refer to *Section 20.4.12* and *Section 20.4.13* for details on clock enabling.

20.4.11 Peripheral allocation inside RCC

Each CPU can allocate any peripheral and hence control its kernel and bus interface clock. A peripheral is allocated when dedicated PERxEN is set in RCC_PERxCFGR (PERx is the peripheral x).

A peripheral is deallocated when dedicated PERxEN is reset in RCC_PERxCFGR.

According to the product RIF policy (see *Section 20.7* for details), each CPU peripheral domain is created based on the following criteria (see *Table 115* for details):

- RIF peripheral configuration

  Filters which CPU is allowed to allocate/deallocate the peripheral. Each RIF peripheral configuration is propagated to the RCC via perxrifcfgr signals.
- RCC RIF resource configuration

  Concerns peripherals that can be used simultaneously by both CPUs and are mainly the RIF-aware peripherals and the internal memories (see *Table 138*).

  The RCC RIF resource configuration must be set in “Shared between different masters with semaphore protection” to allow these peripherals to be shared by both CPUs.

Warning: Any peripheral configured in single-access mode must be deallocated to become free for any other CPU. If not, any RIF configuration change does not work for any new allocation.

Nevertheless, any peripheral listed in the table below can be allocated by one CPU when the related RCC RIF resource configuration is set in “single-access” mode. The application must program the RIF peripheral configuration accordingly.

Example: If the application wants HPDMA1 to be used only by CPU1, the following steps are needed:

1. Set HPDMA1 RIF configuration in CPU1 single-access mode.
2. Set RCC HPDMA1 RIF configuration in CPU1 single-access mode (CFEN = 1, SEM_EN = 0 and SCID = CPU1 ID in RCC_R83CIDCFGR).

Table 138.Shareable peripherals list

Category	List
Peripherals	DDR(1), RCC, PWR, EXTI, SYSCFG, IPCC1, HPDMA1, HPDMA2, HPDMA3, GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH, GPIOI, GPIOZ, FMC, BSEC, ETR, DBGMCU, RTC
Memories	SYSRAM, OCTOSPI1, SRAM1, RETRAM, BKPSRAM

1. DDR can be set shareable according to DDRSHR in RCC_DDRITFCFGR.

When the RIF peripheral configuration is set in legacy mode (CFEN = 0):

- The peripheral is allocated to the first allowed CPU which has set the enable bit. The application must manage the peripheral allocation/deallocation in case of shareable peripheral.

When the RIF resource configuration is set in legacy mode (CFEN = 0):

- The resource is allocated to the latest allowed CPU which has set the enable bit except for the ones listed in *Table 138*. Once the resource is allocated, it is also available for the other CPU (clocking depends on the system state).

PERxEN is used by the hardware to provide the kernel and bus clocks according to the CPU state for peripherals not listed in *Table 138*, whereas it is based on the system state for peripherals set as shareable in *Table 138*.

Clock enabling delays

In order to avoid spurs, the clock gating logic synchronizes the enable command (coming generally from a clock request or PERxEN) with the selected clock.

- A maximum delay of 2 periods of the enabled clock may occur between the enable command and the first rising edge of the clock. The enable command can be the rising edge of PERxEN, or a clock request asserted by a peripheral.
- A maximum delay of 1.5 periods of the disabled clock may occur between the disable command and the last falling edge of the clock. The disable command can be the falling edge of PERxEN, or a clock request released by a peripheral.

Note: Both kernel and bus interface clocks are affected by this resynchronization delay.

In addition, the clock enabling delay may increase if the application enables for the first time, a peripheral which is not located into its domain (the subsystem where the peripheral is located may not have its subsystem clock ready). The subsystem clock must be switched on before the application can use this peripheral.

For example if CPU2 is enables a peripheral located into the AXI subsystem, while the CPU1 is in CStop, the RCC has to provide a clock to the AXI subsystem.

In order to allocate properly a peripheral, the following sequence must be respected:

1. Enable the peripheral clocks (allocate the peripheral) by writing PERxEN = 1.
2. Read back the RCC_PERxCFGR register. This operation can be considered as a dummy read.
3. The peripheral can now be used.

Note: When the bus interface clock is not active, the read/write accesses to the peripheral registers are not supported. A read access results in reading invalid data. A write access is ignored, and do not create any bus errors.

20.4.12 Peripheral clock gating control

As mentioned previously, each peripheral requires a bus interface clock, named ck_icn_p_perx or ck_icn_s_perx (for peripheral x). This clock is an APB, AHB or AXI clock, according to which bus the peripheral is connected.

The clock source used as bus interface clock for peripherals located into the CPU1 domain can be one of the *Table 121* (such as ck_icn_hs_mcu, ck_icn_ls_mcu, ck_icn_sdmmc, or ck_icn_hsl) depending on the bus connected to each peripheral.

Some peripherals (such as SAI, or UART) need also a dedicated clock for their communication interface. This clock is generally asynchronous with respect to the bus interface clock, and is named kernel clock (ck_ker_perx).

Bus interface and kernel clocks can both be gated according to several conditions detailed hereafter.

As shown in the *Figure 107*, the enabling of the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN, PERxLPEN bits in RCC_PERxCFGR
  - PERxEN represents the peripheral enable (allocation) bit for the CPU.
  - PERxLPEN represents the peripheral low-power enable bit.
- cpu1_mode, cpu2_mode (see *Section 20.4.13* for details)
  - CPU1 state (cpu1_mode) is a logic combination of CPU states (WFI or WFE) and STPREQ_P0 bit (see *Table 141* for details).
  - For CPU2 (cpu2_mode), the state depends on the low-power core setting (see *CPU2 clock gating*).
- sys_mode
  - System mode (sys_mode) is used to filter the clock request of autonomous peripherals.
- kernel clock request (req_ck_ker_perx) of the peripheral itself (when feature available)
- bus clock request (req_ck_bus_perx) of the peripheral itself (when feature available) (autonomous peripherals).
- flexclkgen cross bar channel selection (XBARxSEL in RCC_XBARxCFGR)

Due to jitter constraints, the final clock gating stage for bus and/or kernel clocks is not inside the RCC but at the peripheral boundary. The RCC provides only the enable signal for the following peripherals: USBH, OTG, USB2PHY1 and USB2PHY2 (see *Section 20.4.11* for more details).

Figure 107. Peripheral clock enable logic details

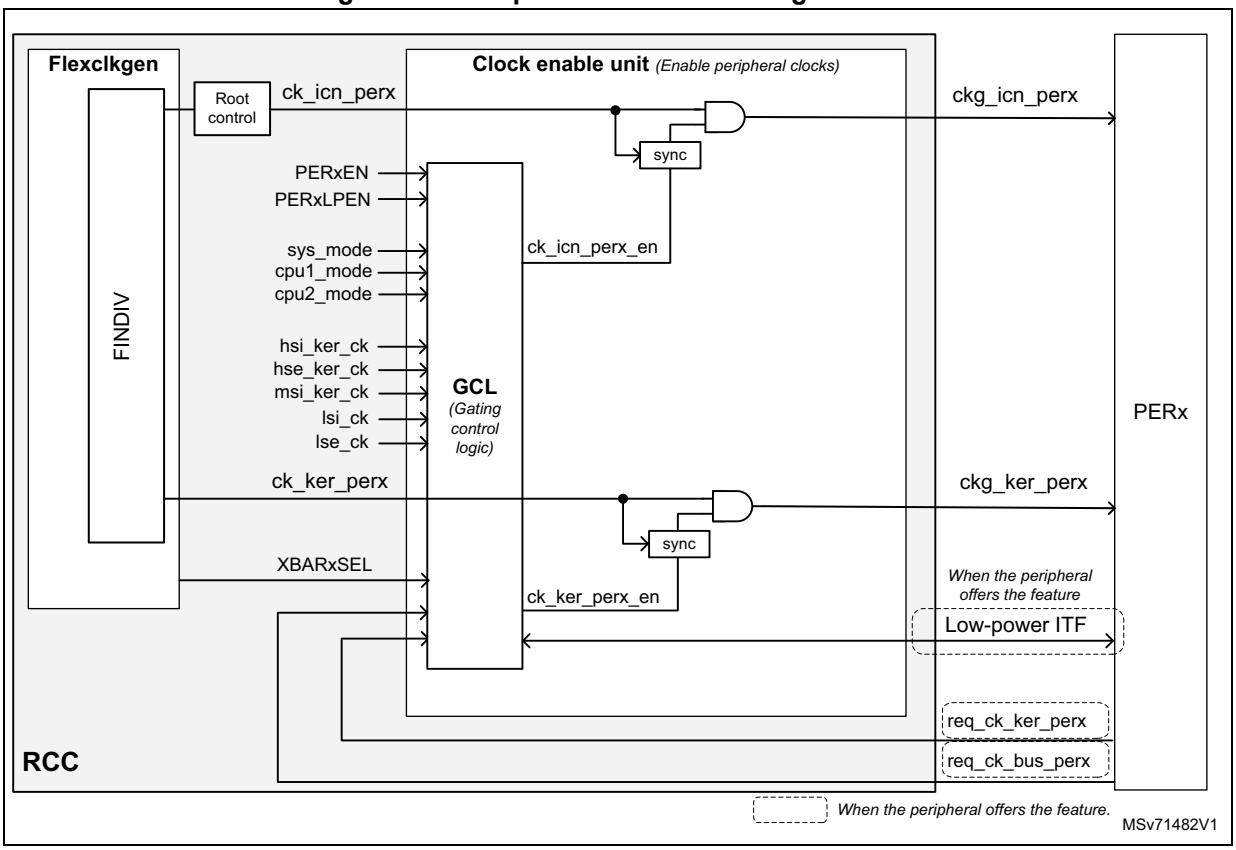


Table 139 and *Table 140* detail the enabling logic of most peripheral clocks for peripherals allocated by CPUs.

Table 139.Peripheral bus clock enabling logic

PERxEN	PERxLPEN	req_ck_bus_perx(1)	CPUx(2)	System mode	ck_icn_perx_en	Comments
0	X	X	X	X	0	No clock provided to the peripheral (PERxEN = 0)
1	X	X	CRun	Run	1	Bus interface clock provided to the peripheral (CPUx in CRun and PERxEN = 1)
1	0	X	CSleep		0	No clock provided to the peripheral (CPUx in CSleep and PERxLPEN = 0)
1	1	X			1	Bus interface clock provided to the peripheral (CPUx in CSleep and PERxLPEN = 1)

Table 139.Peripheral bus clock enabling logic (continued)

PERxEN	PERxLPEN	req_ck_bus_perx[1]	CPUx[2]	System mode	ck_icn_perx_en	Comments
1	0	X	CStop	Run, Stop or LP-Stop	0	No clock provided to the peripheral (CPUx in CStop and PERxLPEN = 0).
1	1	0			0	No clock provided to the peripheral (CPUx in CStop, system in Run, Stop or LP-Stop mode, and req_ck_bus_perx = 0)
1	1	1			1	Bus interface clock provided to the peripheral (CPUx in CStop, system in Run, Stop or LP-Stop mode, and PERxEN = PERxLPEN = req_ck_bus_perx = 1) In system Stop or LP-Stop, HSI is the source clock
1	1	0	CStop	LPLV-Stop	0	No clock provided to the peripheral (system in LPLV-Stop and no bus clock request pending)
1	1	1			1	Bus interface clock provided to the peripheral (system in LPLV-Stop and req_ck_bus_perx = 1)

1. See *Section 19.5.2: Low-power batch acquisition mode (LPBAM)* for the list of supported autonomous peripherals. All enabled autonomous peripherals are clocked when at least one req_ck_bus_perx is active.
2. CPUx = CPU1, CPU2 (see *Section 20.4.14* for details).

As a summary, the bus interface clock is provided to the peripherals in the following conditions:

- when the CPU to which the peripheral is allocated is in CRun
- when the CPU is in CSleep, with PERxLPEN = 1
- when the CPU is in CStop, system in Run, Stop or LP-Stop mode, with PERxLPEN = 1, and the peripheral generates a bus clock request
- when the CPU is in CStop, system in LPLV-Stop mode, with PERxLPEN = 1, and the peripheral generates a bus clock request

Table 140.Peripheral kernel clock enabling logic

PERxEN	PERxLPEN	XBARxSEL	req_ck_ker_perx	CPUx[1]	System mode	ck_ker_perx_en	Comments
0	X	X	X	X	X	0	No clock provided to the peripheral (PERxEN = 0)
1	X	X	X	CRun	Run	1	Kernel clock provided to the peripheral (CPUx in CRun and PERxEN = 1)
1	0	X	X	CSleep		0	No clock provided to the peripheral (CPUx in CSleep and PERxLPEN = 0)
1	1	X	X			1	Kernel clock provided to the peripheral (CPUx in CSleep and PERxLPEN = 1)

Table 140.Peripheral kernel clock enabling logic (continued)

PERxEN	PERxLPEN	XBARxSEL	req_ck_ker_perx	CPUx(1)	System mode	ck_ker_perx_en	Comments
1	1	no lsi_ck and no lse_ck and no hsi_ker_ck and no msi_ker_ck and no hse_ker_ck	X	CStop	Run, Stop or LP-Stop	0	No clock provided to the peripheral (CPUx in CStop and lse_ck, lsi_ck, hsi_ker_ck, msi_ker_ck or hse_ker_ck not selected)
1	1	lsi_ck or lse_ck	X			1	Kernel clock provided to the peripheral (PERxEN = PERxLPEN = 1 and lsi_ck or lse_ck selected) The application must enable LSE or LSI if requested.
1	1	hsi_ker_ck or msi_ker_ck or hse_ker_ck	1			1(2)	Kernel clock provided to the peripheral (req_ck_ker_perx = PERxEN = PERxLPEN = 1 and hsi_ker_ck, msi_ker_ck or hse_ker_ck selected) If the oscillators are off, the RCC enables them when a kernel clock request occurs.
1	1	hsi_ker_ck or msi_ker_ck or hse_ker_ck	0(3)			0	No clock provided to the peripheral (CPUx in CStop, and no kernel clock request pending)
1	1	lsi_ck or lse_ck or msi_ker_ck	0(3)	CStop	LPLV-Stop	0	No clock provided to the peripheral (no kernel clock request pending)
1	1		1			1	Kernel clock provided to the peripheral (PERxLPEN = 1, lsi_ck or lse_ck or msi_ker_ck selected and req_ck_ker_perx = 1)

1. CPUx = CPU1 or CPU2 (see *Section 20.4.14* for details).
2. The DTS does not support the kernel clock request, but the kernel clock is still delivered in CStop.
3. For MDF1 the kernel clock is provided when PERxEN = PERxLPEN = 1 if selected from a PLL, see *Table 125* footnote (4) for oscillator cases. For LPTIM1/2/3/4/5, the kernel clock is provided when lsi_ck or lse_ck is selected.

Note: The kernel clock request generated by the peripheral, enables the corresponding oscillator only if XBARxSEL[3:0] in RCC_XBARxCFGR selects an oscillator: HSI, MSI or HSE.

As a summary, the kernel clock is provided to the peripherals in the following conditions:

- when the CPU to which the peripheral is allocated is in CRun
- when the CPU is in CSleep, with PERxLPEN = 1
- when the CPU is in CStop, with PERxLPEN = 1, the peripheral generates a kernel clock request, and the selected clock is hsi_ker_ck or msi_ker_ck or hse_ker_ck
- when the CPU is in CStop, with PERxLPEN = 1, and the kernel source clock of the peripheral is lse_ck or lsi_ck
- when the CPU is in CStop, with PERxLPEN = 1, the peripheral generates a kernel clock request, and the selected clock is lsi_ck, lse_ck or msi_ker_ck

Note: Only some peripherals are able to request the kernel clock: I2C, I3C, SPI, U(S)ART. This feature gives to the peripheral the capability to wait for an event with an optimal power consumption.

20.4.13 CPUx and interconnect clocks gating

CPU1 clock gating

The CPU1 embeds one core and its own PLL (PLL1) provides all local clocks. The RCC provides only the PLL1 source reference clock (ck_pll1_ref), and the PLL1 bypass reference clock (ck_cpu1_ext2f).

When the CPU1 core goes to WFI, it is set in low-power state by gating its clocks, except for the functions used to wake up the core from WFI.

When the CPU1 core goes to WFE, it is set in low-power state by gating its clocks, except for the functions used to wake up the core from WFE.

When the CPU1 core is in WFI, the L2 memory system logic can also enter in WFI, and the cpu1_standbywfi_l2 signal is asserted. More precisely, when the CPU1 core is in WFI, the L2 memory system completes the pending transactions, enters in WFI, and cpu1_standbywfi_l2 is asserted. If CStop has been requested via STPREQ_P0 bit, the RCC disables ck_pll1_ref, ck_cpu1_ext2f clocks, and the AXI interface clock (ck_icn_m_cpu1).

The table below details the CPU1 modes according to the L2 memory system logic state (WFI), and RCC register values.

Table 141. CPU1 mode details(1)

STPREQ_P0	L2 memory logic in WFI	CPU1 mode
-	N	CRun
0	Y	CSleep
1	Y	CStop

1. "-" means don't care.

The CPU1 can go in an enhanced CSleep when th ESLPREQ is set in RCC_C1SREQSETR. When this bit is set and STPREQ_P0 bits combination is 0, PLL1 is powered down and bypassed (ck_pll1_ref is also disabled) each time cpu1_standbywfi_l2 is asserted. ck_cpu1_ext2f clock must be used to be able to leave this mode.

When CPU1 is in CStop, PLL1 is also powered down when the system goes in one of Stop1 modes without any software assistance. If the system goes in one of Stop2 modes, the application software is required to disable PLL1 first, before putting the core in WFI state.

PLL1 disable application software procedure is the following:

1. Set CA35SS_SSC_CHGCLKREQ.ARM_CHGCLKREQ.
2. Poll CA35SS_SSC_CHGCLKREQ.ARM_CHGCLKACK bit until it is 1.
3. Clear CA35SS_SSC_PLL_ENABLE.NRESET_SWPLL.
4. Clear CA35SS_SSC_PLL_ENABLE.PLL_EN.
5. Poll CA35SS_SSC_PLL_ENABLE.LOCKP until it is 0.

Figure 108. CPU1 low-power interface

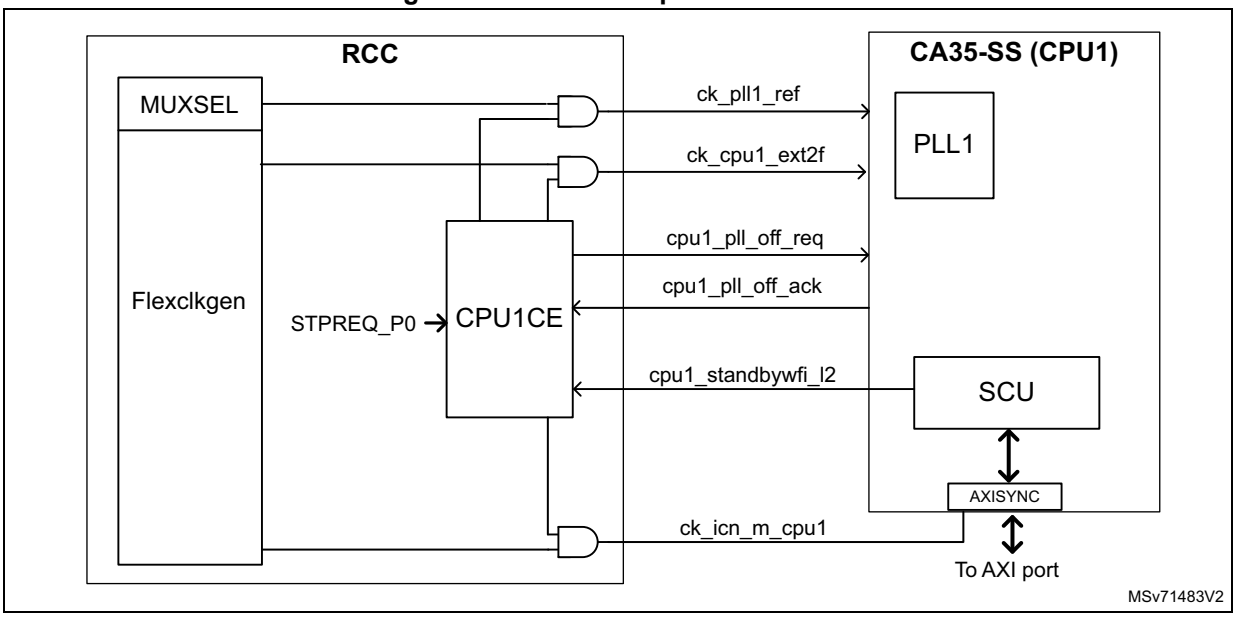
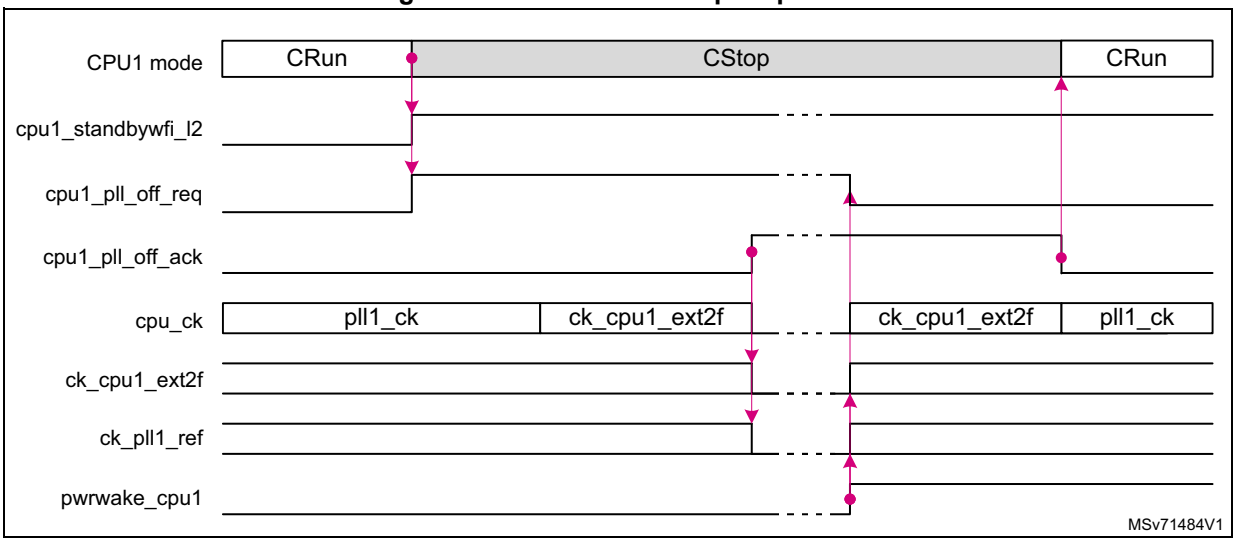


Figure 109. CPU1 PLL1 stop sequence



CPU2 clock gating

CPU2 main clock (ck_cpu2) is sourced from ck_icn_hs_mcu clock.

CPU2 support two low-power states: sleep (SLEEPING output set high) and deepsleep (SLEEPING and SLEEPDEEP outputs set high).

CPU2 sleep state is equivalent to CSleep. CPU2 deepsleep state is equivalent to CStop for RCC. CPU2 main clock is only gated when it is in CStop.

Bus matrix and bridge clock gating

Bridge and bus matrix clocks are gated according to CPUx mode, and to the peripheral allocated to these CPUx. *Table 142* shows in which conditions the RCC gates the bus matrix and bridge clocks (see *Section 20.4.12* for information about convention naming).

Table 142. Bus clock enable details for ck_icn_ahbx with x = 2,5,6 and ck_icn_apby with y = 1,2,3,4,5,dbg

CPU1 (1)	CPU2 (1)	cpu1_peren_ahbx(apby)(2)	cpu1_perlpen_ahbx(apby)(2)	cpu2_peren_ahbx(apby)(2)	cpu2_perlpen_ahbx(apby)(2)	req_ck_bus_perz	ck_icn_ahbx or ck_icn_apby	Description
CRun	Any	1	X	X	X	X	Clocked	The clock is generated when one of the following conditions is true: – CPU1 in CRun with at least one peripheral on AHBx (APBy) allocated (at least one PERxEN = 1) – CPU1 in CSleep with at least one peripheral on AHBx (APBy) allocated, and with PERxEN = PERxLPEN = 1 – CPU2 in CRun with at least one peripheral on AHBx (APBy) allocated (least one PERxEN = 1 – CPU2 in CSleep with at least one peripheral on AHBx (APBy) allocated, and with PERxEN = PERxLPEN = 1 – both CPU1 and CPU2 in CStop with at least one autonomous peripheral on AHBx (APBy) allocated, with PERxEN = PERxLPEN = 1, and with its bus clock request active (req_ck_bus_perz = 1)
CSleep		1	1	X	X	X		
Any	CRun	X	X	1	X	X		
	CSleep	X	X	1	1	X		
CStop	CStop	1	1	X	X	1		
		X	X	1	1	1		
Any	Any	0	X	0	X	X	Gated	In all other cases, the clock is gated.
	CSleep	0	X	X	0	X		
	CStop	0	X	X	X	X		
CSleep	Any	X	0	0	X	X		
	CSleep	X	0	X	0	X		
	CStop	X	0	X	X	X		
CStop	Any	X	X	0	X	X		
	CSleep	X	X	X	0	X		
	CStop	1	1	X	X	0		
		X	X	1	1			

1. ‘Any’ means CRun, CSleep, or CStop.
2. cpu1_peren_ahbx(apby) is set to 1 when at least one peripheral connected to AHBx (APBy) is allocated to CPU1.
   cpu1_perlpen_ahbx(apby) is set to 1 when at least one peripheral connected to AHBx (APBy) is allocated to CPU1, with its PERxLPEN set to 1.
   cpu2_peren_ahbx(apby) is set to 1 when at least one peripheral connected to AHBx (APBy) is allocated to CPU2.
   cpu2_perlpen_ahbx(apby) is set to 1 when at least one peripheral connected to AHBx (APBy) is allocated to CPU2, with its PERxLPEN set to 1.

20.4.14 Mass storage interface clocking

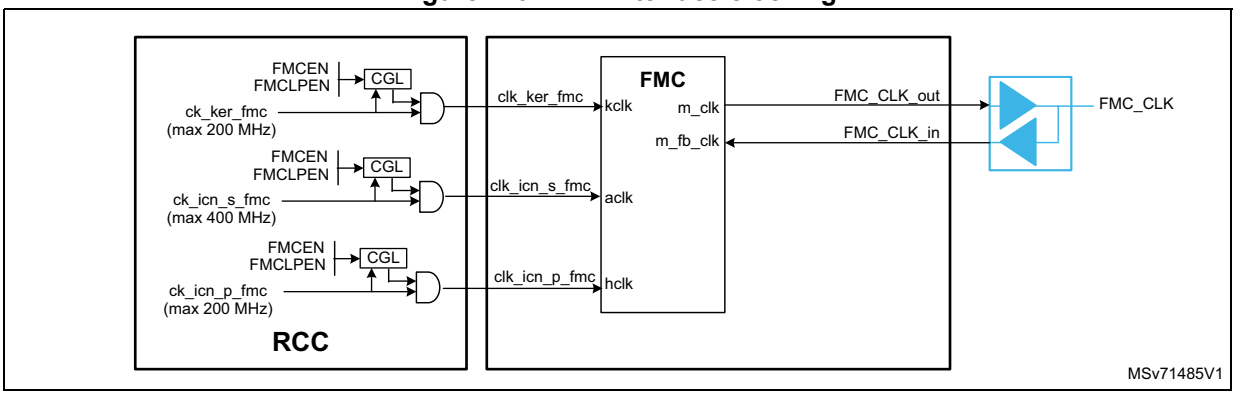
One programmable delay (DLYB) is used for each SDMMC and the OCTOSPI interface. These programmable delays are used:

- to tune the RX sampling clock phase for SDMMC1/2/3, OCTOSPI1 controllers
- to tune the TX launch clock phase for OCTOSPI1, and thus to meet setup/hold constraints of TX signals versus the OCTOSPI memory clocks

These programmable delays are based on a DLL that compensates voltage and temperature variations. This DLL features 32 taps and covers the frequency range from 50 MHz to 208 MHz. The clock on which the DLL locks is provided by SDMMC or OCTOSPI controllers. This clock is at the same frequency as the memory clock, and it is free running. The programmable delays control bits are described in system configuration controller (SYSCFG) section. More details in Mass storage interfaces section.

FMC interface - Clocking schematic

Figure 110. FMC interface clocking



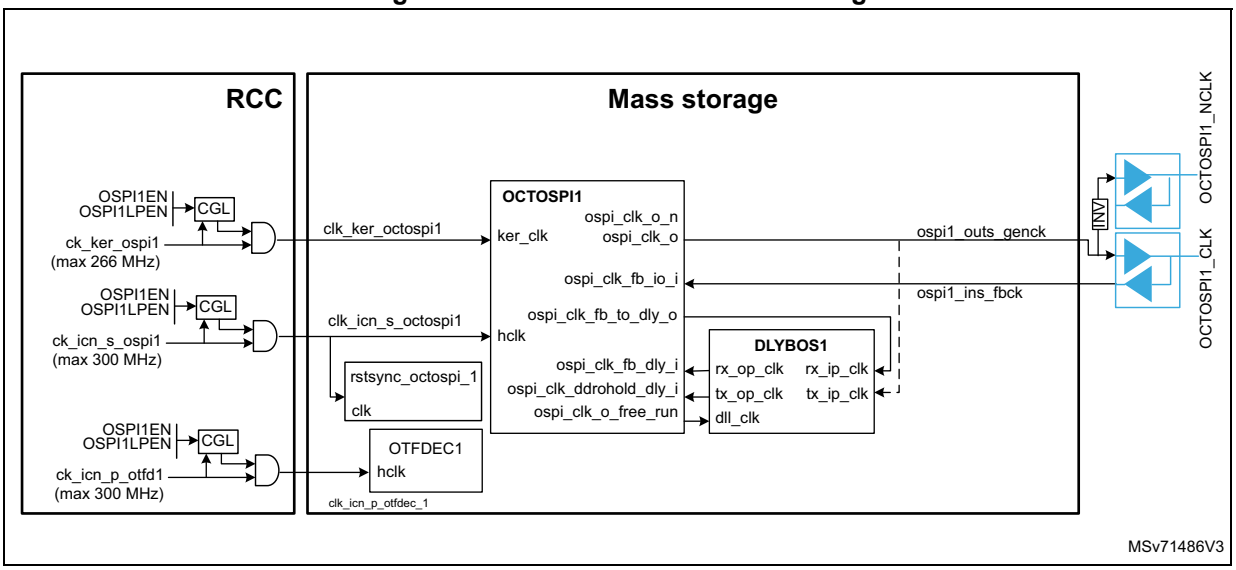
The FMC controller receives its kernel, AHB and AXI clocks from the RCC.

Table 143. FMC interface clocks from RCC

Subsystem clock signal name	Peripheral	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
clk_ker_fmc	FMC	KCLK	Input	ck_ker_fmc	200 MHz	kernel clock
clk_icn_s_fmc		ACLK	Input	ck_icn_s_fmc	400 MHz	AXI clock
clk_icn_p_fmc		HCLK	Input	ck_icn_p_fmc	200 MHz	AHB clock

OCTOSPI interface - Clocking schematic

Figure 111. OCTOSPI interface clocking



The RCC provides kernel and bus clocks to OCTOSPI1 and OTFDEC1. DLYBOS1 provide delayed feedback clocks, and delayed TX launch clocks to OCTOSPI1.

Table 144. OCTOSPI clocks from RCC

Subsystem clock signal name	Peripheral	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
clk_ker_octospi1	OCTOSPI1	ker_clk	Input	ck_ker_ospi1	266 MHz	kernel clock
clk_icn_s_octospi1	OCTOSPI1	HCLK	Input	ck_icn_s_ospi1	300 MHz	AHB clock
clk_icn_p_otfdec_1	OTFDEC1	HCLK	Input	ck_icn_p_otfd1	300 MHz	AHB clock

Table 145. OCTOSPI interface clocks from DLYBOS1

Peripheral	Clock signal name	Clock direction	Max frequency	Description
DLYBOS1	rx_op_clk	Output	200 MHz	Delayed RX sampling clock
DLYBOS1	tx_op_clk	Output	200 MHz	Delayed TX launch clock

Details of the OCTOSPI interface clocking

For OCTOSPI1 interface, the RX sampling strobe can be selected between:

- the memory clock after loopback at pin level (OCTOSPI ospi_clk_fb_io_i input)
- a delayed version of this loopback clock (OCTOSPI ospi_clk_fb_dly_i input)
- the DQS signal provided by the memory device (OCTOSPI afi_dqs input)
- a delayed version of DQS (OCTOSPI ospi_clk_fb_dly_i input)

For OCTOSPI1 interface, the delay on loopback clock or DQS relies on a DLL. This DLL is used to tune the phase of the RX sampling strobe, so that it is positioned in the middle of the

data valid window, and guarantees that this phase remains constant even if voltage or temperature conditions vary.

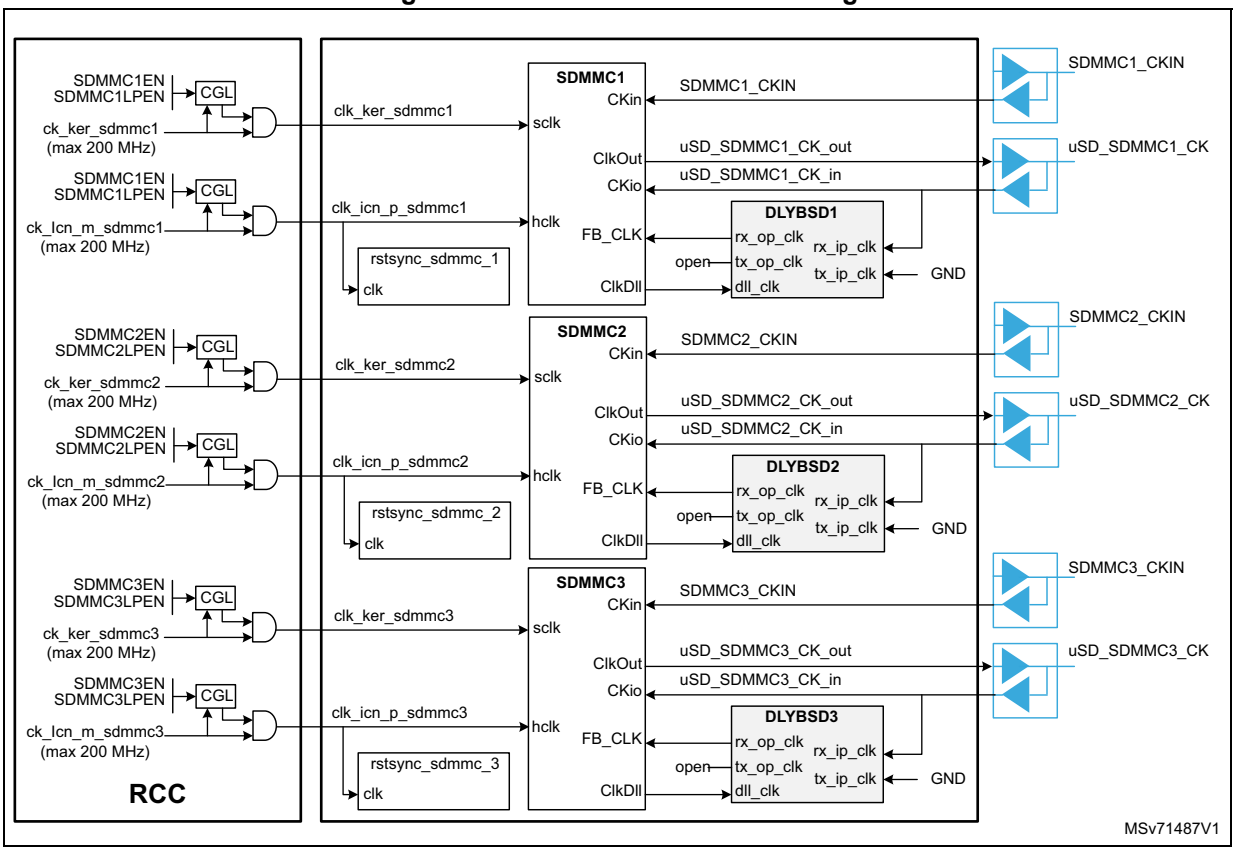
Using this DLL is mandatory at all frequencies for protocols where a DQS is provided by the memory device to qualify read data. This DQS is edge-aligned with data at the output of the memory device, and must be shifted to enable data safe sampling. For memory frequencies above 50 MHz, the DLL is used in lock mode. For memory frequencies below 50 MHz, the DLL is used in bypass mode (pure delay).
For non DQS based protocols, using the DLL is recommended when memory frequency is greater than 100 MHz (STR), or 66 MHz (DTR). This is optional for lower speeds where using the loopback clock is sufficient. The DLL must not be used in lock mode for speed modes below 50 MHz.

A delay relying on a DLL is also implemented for the TX path, with the following delay use cases:

- If operating in STR mode, data are generated by the OCTOSPI on the falling edge of the memory clock. This ensures by construction an optimal setup and hold timing versus the rising edge of the memory clock. The TX delay is not used.
- If operating in DTR mode with a kernel clock that is a multiple of the memory clock, DHQC in OCTOSPI_TCR must be set to 1. This bit is used to shift the data versus memory clock edges (shift by T/4 if kernel clock frequency is twice the memory clock frequency). The TX delay is not used.
- If operating in DTR mode with a kernel clock frequency equal to the memory clock frequency, the TX delay is required to shift the data versus the memory clock edges. This delay is used to shift the TX launch clock (OCTOSPI ospi_clk_ddrohold_dly_i input).

SDMMC interface - Clocking schematic

Figure 112. SDMMC interface clocking



The RCC provides the kernel and bus clocks to the SDMMC1/2/3. DLYBSD1/2/3 provide delayed feedback clocks for RX sampling to the SDMMC1/2/3.

Table 146. SDMMC clocks from RCC

Subsystem clock signal name	Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
clk_ker_sdmmc1	SDMMC1	SClk	Input	ck_ker_sdmmc1	200 MHz	kernel clock
clk_icn_p_sdmmc1	SDMMC1	HCLK	Input	ck_icn_m_sdmmc1	200 MHz	AHB clock
clk_ker_sdmmc2	SDMMC2	SClk	Input	ck_ker_sdmmc2	200 MHz	kernel clock
clk_icn_p_sdmmc2	SDMMC2	HCLK	Input	ck_icn_m_sdmmc2	200 MHz	AHB clock
clk_ker_sdmmc3	SDMMC3	SClk	Input	ck_ker_sdmmc3	200 MHz	kernel clock
clk_icn_p_sdmmc3	SDMMC3	HCLK	Input	ck_icn_m_sdmmc3	200 MHz	AHB clock

Table 147. SDMMC interface clocks from DLYBSD1/2/3

Peripheral	Clock signal name	Clock direction	Max frequency	Description
DLYBSD1	rx_op_clk	Output	200 MHz	Delayed RX sampling clock
DLYBSD2	rx_op_clk	Output	200 MHz	Delayed RX sampling clock
DLYBSD3	rx_op_clk	Output	200 MHz	Delayed RX sampling clock

Details of the SDMMC interface clocking

For all SDMMC1/2/3 interfaces, the RX sampling clock can be selected between:

- the memory clock after loopback at pin level (SDMMC CKio input)
- a delayed version of this loopback clock (SDMMC FB_CLK input)
- the memory clock after loopback in an external level shifter (SDMMC CKin input)

For each SDMMC interface, the delayed loopback clock relies on a DLL. This DLL is used to tune the phase of the RX sampling clock so that it is positioned in the middle of the data valid window (also valid for Cmd), and guarantees that this phase remains constant even if voltage or temperature conditions vary. Using this DLL is mandatory for speed modes where the phase of the data (respectively Cmd) valid window is not guaranteed by the memory device (SDR104, HS200 modes). This is optional for other speed modes where using the loopback clock is sufficient. The DLL must not be used in lock mode for speed modes below 50 MHz.

There is no programmable delay nor DLL implemented for the TX paths of the SDMMC. The SDMMC generates data (respectively Cmd) with the maximum setup/hold timings versus the active edge of the clock sent to the memory device.

20.4.15 Ethernet interface clocking

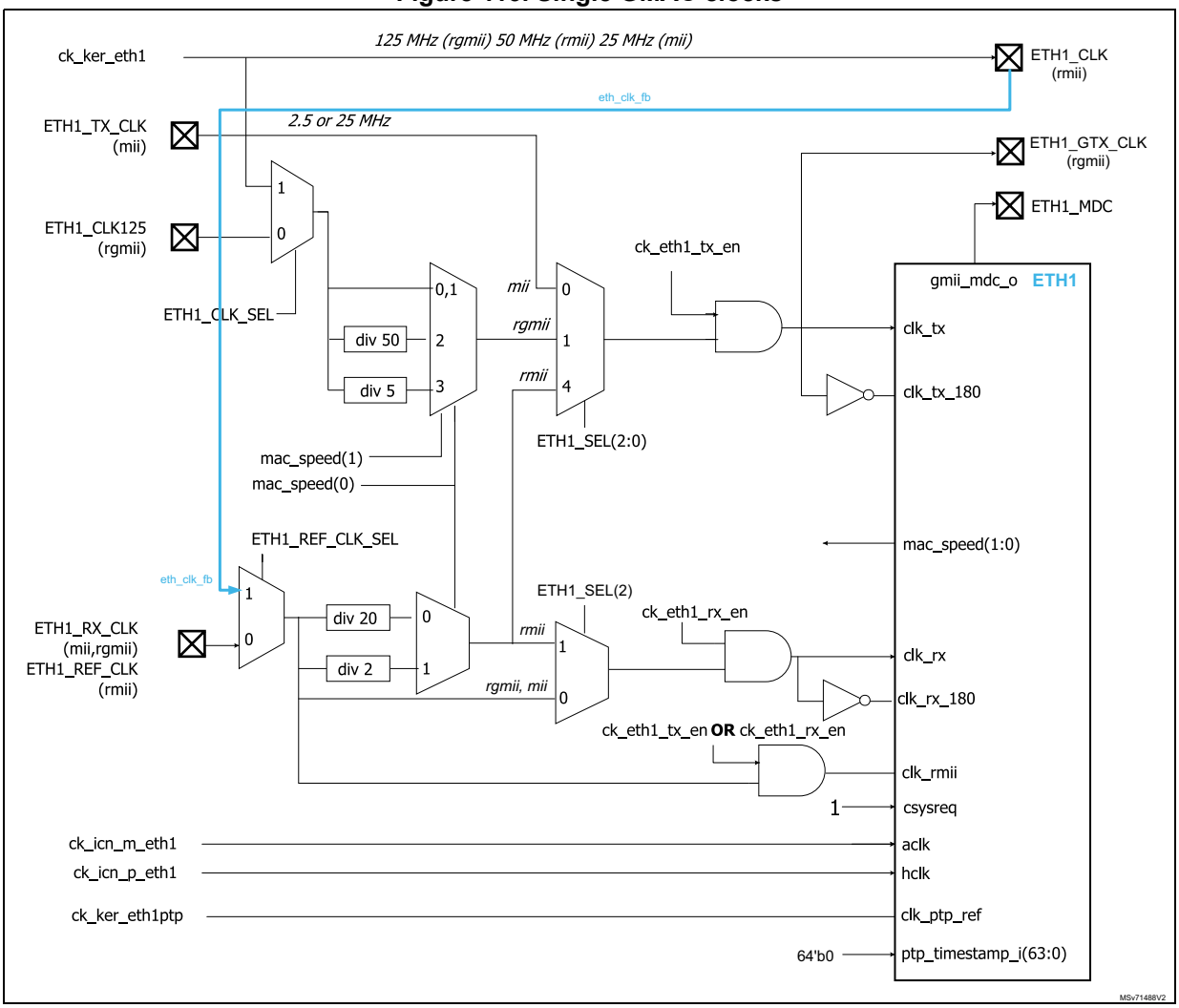
There are two independent GMAC controllers (ETH1 and ETH2) that support MII, RMII and RGMII interfaces.

ETH1 and ETH2 include an MDIO interface. External clock I/Os are multiplexed to support internal or external clock generation. ETH1 and ETH2 support TSN.

Clocking schematic

Each GMAC has the same (but independent) clock management. The clocking for a single GMAC is shown in the figure below.

Figure 113. Single GMAC clocks



The RCC provides all the clocks except from the I/O inputs (ETH1_CLK125, ETH1_TX_CLK, ETH1_RX_CLK). the Ethernet clocks are also available as outputs (ETH1_CLK, ETH1_GTX_CLK) when the device is the clock master.

The main clock for the Ethernet is generated by the RCC (ck_ker_eth1) or externally (ETH1_CLK125 or ETH1_REF_CLK).

Clock signals

Table 148. Ethernet clocks from RCC

Subsystem clock signal name	Peripheral	Clock signal name	Clock direction	RCC clock	Max frequency	Clock description
clk_eth1_hclk	ETH1 controller	hclk	Input	ck_icn_p_eth1	150 MHz	AHB register access
clk_eth1_ axi_master	ETH1 controller	aclk	Input	ck_icn_m_eth1	300 MHz	AXI data master
clk_ker_eth1	ETH1 controller	(multiple divided)	Input	ck_ker_eth1	125 MHz	Internal GMAC clock
clk_eth1_ptp_ref	ETH1 controller	clk_ptp_ref	Input	ck_ker_eth1ptp	200 MHz	PTP reference clock

In the table below, signals and I/Os for ETH2 are not shown, but repeat these called “_eth1” or “ETH1” with “_eth2” or “ETH2”.

Table 149. Ethernet clocks to/from I/Os

Subsystem clock pin name	Peripheral	Clock signal name	Clock direction	Max frequency	Clock description
ETH1_CLK125	ETH x controller	-	Input	125 MHz	Optional RGMII external reference (fixed frequency)
ETH1_TX_CLK		-	Input	25 MHz	MII external TX clock
ETH1_RX_CLK/ ETH1_REF_CLK		-	Input	125 MHz	MII or RGMII external RX clock, RMII external reference
ETH1_CLK		-	Output	125 MHz	To external PHY (fixed frequency)
ETH1_GTX_CLK		clk_tx_i	Output	125 MHz	To external PHY (dynamic frequency)
ETH1_MDC		gmii_mdc_o	Output	hclk/4	Clock for MDIO

All RGMII I/Os must support 125 MHz dual-edge toggling. Only 1.8 V and 3.3 V I/O signaling is supported (V_{DD} common to other GPIOs). 1.5 V support is not required as most PHYs also support 1.8 V. This means that the RGMII v2.0 is supported except for HSTL 1.5 V.

Ethernet I/O use

Table 150. Active I/O per mode (MII, RMII, RGMII)

Direction	MII	RMII	RGMII	Comments
Input/output	ETH_CLK	ETH_CLK	ETH_CLK	125/50/25 MHz to PHY (static)
Output	-	-	GTX_CLK	125/25/2.5 MHz in RGMII (dynamic)
Input	-	-	ETH_CLK125	125 MHz from PHY (static)
Input	RX_CLK	REF_CLK	RX_CLK	125/25/2.5 MHz, or 50 MHz RMII
Input	RX_DV	CRS_DV	RX_CTL	-

Table 150. Active I/O per mode (MII, RMII, RGMII) (continued)

Direction	MII	RMII	RGMII	Comments
Input	RX_ER	-	-	-
Input	COL	-	-	-
Input	CRS	-	-	-
Input	RXD0	RXD0	RXD0	DDR (except MII)
Input	RXD1	RXD1	RXD1	-
Input	RXD2	-	RXD2	-
Input	RXD3	-	RXD3	-
Input	TX_CLK	-	-	25/2.5 MHz in MII
Input	TX_EN	TX_EN	TX_CTL	-
Output	TXD0	TXD0	TXD0	DDR (except MII)
Output	TXD1	TXD1	TXD1	-
Output	TXD2	-	TXD2	-
Output	TXD3	-	TXD3	-
Output	MDC	MDC	MDC	2.5 MHz or faster
Input/output	MDIO	MDIO	MDIO	-

Description

The two controllers have separate register programming ports with independent clocks. Each controller also has a master port for data transfers, clocked by ck_icn_m_eth1 or ck_icn_m_eth2.

The RCC clock enable ck_ker_eth1_en gates the main Ethernet clock. There are also RCC clock enable for transmit (ck_eth1_tx_en) and receive (ck_eth1_rx_en) clocks, and an interconnect clock enable (ck_icn_eth1_en) for aclk and hclk.

The application can control the gating according to the CPUx modes (CRun, CSleep). It can also control the gating of clocks coming from ETH1_RX_CLK and ETH1_TX_CLK I/Os in CStop and Stop modes (see ETH1STPEN in RCC_ETH1CFGR).

The main Ethernet clock provided by the RCC is available on ETH1_CLK I/O. The clock selection for the RX and TX paths is controlled via the SYSCFG.

ETH1_SEL[2:0] in RCC_ETH1CFGR must be set to match the interface type used.

MII mode

The transmit clock is received from the external PHY via ETH1_TX_CLK I/O. The clock frequency is 2.5 or 25 MHz. The receive clock is received from the external PHY via ETH1_RX_CLK I/O. The clock frequency is 2.5 or 25 MHz. The ETH1_CLK output can be used to provide a reference clock (25 MHz) to the external PHY.

RGMII mode

The transmit clock can be selected between an internal clock provided by the RCC, or the clock received on the ETH1_CLK125 I/O. This selection is controlled via ETH1_CLK_SEL

in SYSCFG_ETH1CR. The clock frequency can be adjusted dynamically to 2.5, 25 or 125 MHz according to the mac_speed_o[1:0] signal provided by the ETH1 controller

The receive clock is coming directly from ETH1_RX_CLK I/O. The clock frequency is 2.5, 25, or 125 MHz. The dynamic transmit clock is provided to the PHY via ETH1_GTX_CLK I/O.

A constant reference clock of 125 MHz (50% duty cycle ± 5%) can also be provided to the external PHY via ETH1_CLK I/O.

Note: When the internal clock is provided externally, ck_ker_eth1 must not be gated.

RMII mode

The continuous reference clock (50 MHz) can be internally generated (ck_ker_eth1), or can come from ETH1_REF_CLK I/O. This selection is controlled via ETH1_REF_CLK_SEL in SYSCFG_ETH1CR.

The transmit and receive clock frequency can be adjusted dynamically to 2.5 or 25 MHz according to the mac_speed_o[0] signal provided by the ETH1 controller.

A continuous reference clock (50 MHz) can also be provided to the external PHY via ETH1_CLK I/O. To track the skew on the board when the internal clock is selected, the RMII clock (eth_clk_fb) is actually driven from the ETH1_CLK I/O buffer.

Note: To be continuous, ck_ker_eth1 must not be gated in Stop mode.

Dynamic clock generation

Internal transmit and receive clocks vary dynamically according to the MAC interface speed. The internal transmit clock is also available on ETH1_GTX_CLK I/O.

The ETH1 controller reports the speed of the MAC interface which may change dynamically during normal operation.

Table 151. Ethernet MAC speed encoding

mac_speed_o[1:0]	MAC speed (Mbit/s)
0x	1000
10	10
11	100

Clock multiplexers controlled by mac_speed_o produce a glitch-free output.

Station management clock

ETH1_MDC is the output clock for the station management interface. The MDC clock is a divided version of ck_icn_p_eth1 ("CSR clock"). The clock range (CR) setting in ETH1_MACMDIOAR selects the division ratio.

IEEE 802.3 specifies a maximum MDC frequency of 2.5 MHz, but higher frequencies can be selected if required. 12 MHz is supported by STM32MP21xx devices, but ETH1 supports up to 50 MHz (hclk/4). See *Section 77: Ethernet (ETH): gigabit media access control (GMAC) with DMA controller* section for the available ratios. The same register can be used to add extra clock cycles at the end of an MDIO frame transmission.

20.4.16 CA35SS clocking

Clocks definition

All clocks provided to the CA35SS are controlled from the RCC, and related information to the clocks gating are described in the RCC specification. The table below lists clocks from the RCC and their usage in the CA35SS.

Table 152. Cortex-A35 interface clocks from RCC

RCC clock line	CA35SS instance	Max frequency	Description
ck_cpu1_ext2f	CRMR	400 MHz	Default clock frequency for Cortex-A35 core
ck_pll1_ref	PLL1	64 MHz	Reference clock for Cortex-A35 PLL1
ck_icn_m_cpu1	AXI4 bridge	600 MHz	AXI4 clock for master side of asynchronous bridge
ck_icn_s_gic400	GIC400	400 MHz	Interface clock for GIC400
ck_ker_stgen	STGEN	64 MHz	STGEN timer clock
clk_icn_p_stgen	STGEN	150 MHz	STGEN programming interface clock
ck_icn_p_syscpu1	SYSCFG	150 MHz	Interface clock for system configuration registers

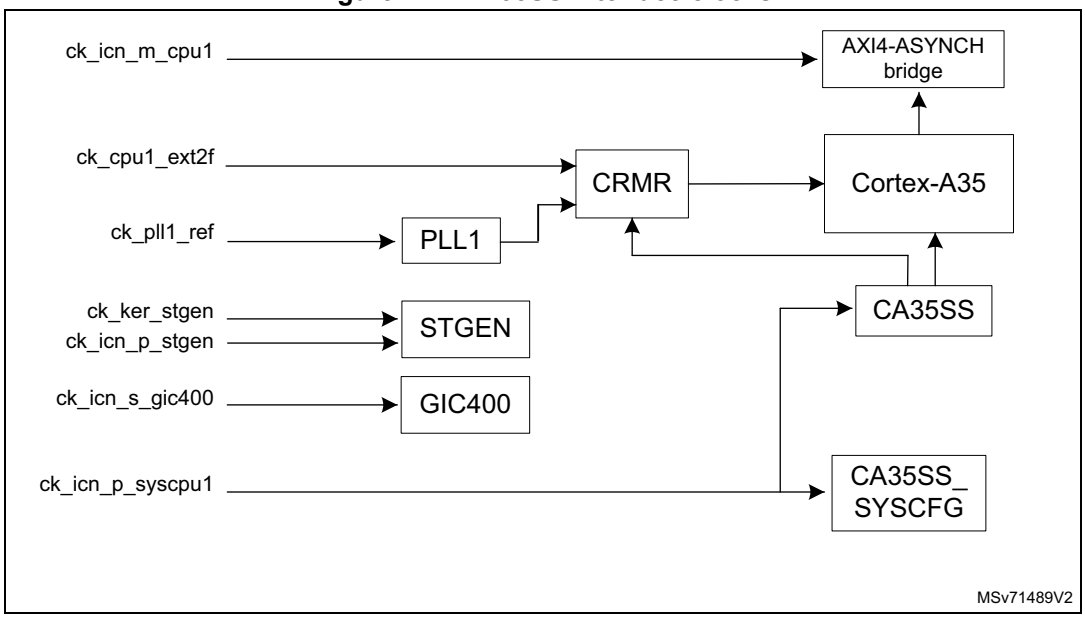
The reference clock for the PLL1 can be selected among three sources (HSI, HSE, MSI) through RCC_MUXSELCFGR register. Four of these clocks have explicit clock gating controlled from RCC registers as follows:

- ck_ker_stgen and clk_icn_p_stgen from RCC_STGENCFGR
- ck_icn_p_syscpu1 from RCC_SYSCPU1CFGR

The others have dynamic clock gating depending on the Cortex-A35 core power states.

Clocking schematic

Figure 114. CA35SS interface clocks



The following clock sources can be selected by CA35SS registers to compute the Cortex-A35 clock, which is the main clock to the Arm core and its caches:

- clkext2f
- clkext2f divided by 2
- PLL1 output clock

The CRMR handles this selection by providing appropriate clocks in a “glitch free” manner.

Note: After initial power-up reset, the dual Cortex-A35 clock is driven from clkext2f (not divided).

PLL1 enable procedure is the following:

1. Clear CA35SS_SSC_PLL_ENABLE.NRESET_SWPLL.
2. Configure PLL parameters in CA35SS_SSC_PLL_FREQ1 and CA35SS_SSC_PLL_FREQ2.
3. Set CA35SS_SSC_PLL_ENABLE.PLL_EN.
4. Poll CA35SS_SSC_PLL_ENABLE.LOCKP until it is 1.
5. Set CA35SS_SSC__PLL_ENABLE.NRESET_SWPLL.
6. Clear CA35SS_SSC_CHGCLKREQ.ARM_CHGCLKREQ.
7. Poll CA35SS_SSC_CHGCLKREQ.ARM_CHGCLKACK until it is 0.

20.4.17 Debug and trace clocking

Main debug and trace clocks are provided according to:

- DAP CDBGPWRUPREQ request.
- RCC_DBGCFGR register (DBGEN, DBGMCUEN, TRACEEN).
- RCC_ETRCFGR and RCC_STMCFGR registers.

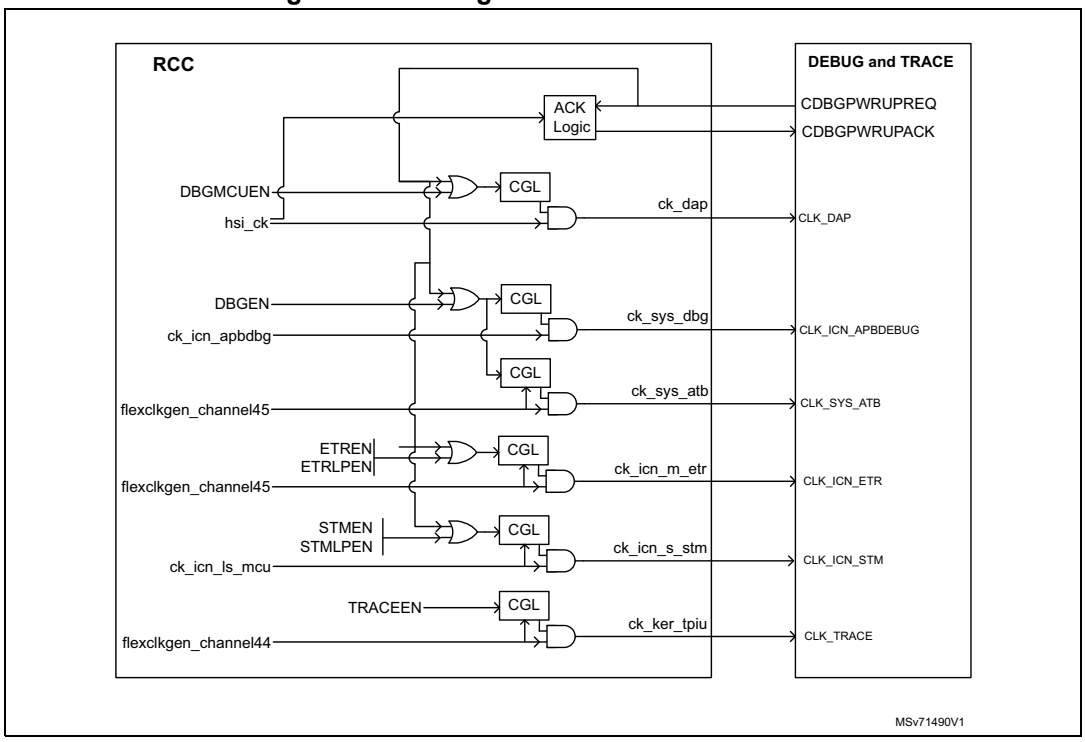
The table below presents the list of the debug and trace external clocks.

Table 153. Debug and Trace subsystem external clocks

RCC clock line	Debug and Trace instance	Instance signal name	Max. frequency	Description
ck_dap	DEBUG	clk_dap	64 MHz	DAP clock domain (DBGMCU included)
ck_sys_dbg	DEBUG	ck_icn_apbdebug	150 MHz	Debug clock domain
ck_sys_atb	DEBUG	clk_sys_atb	400 MHz	ATB clock domain
ck_icn_m_etr	TRACE	clk_icn_etr	400 MHz	ETR AXI master port clock
ck_icn_s_stm	TRACE	clk_icn_stm	150 MHz	STM AXI slave port clock
ck_ker_tpiu	TRACE	clk_trace	200 MHz	Processing clock for trace

Clocking schematic

Figure 115. Debug and Trace external clocks



20.4.18 Camera subsystem clocking

The DCMI and the PSSI have a slow input parallel interface that receives a clock when this interface is used. Both share a clock for their slave interface.

The DCMIPP has a fast input parallel interface that receives a clock when this interface is used. It also has a clock for its internal pixel processing, and its master and slave interface.

The CSI interface requires a hard macrocell which retrieves the clock. This clock is then used for processing.

The table below presents the list of the camera subsystem external clocks.

Table 154. Camera subsystem external clocks

Subsystem clock signal name	Camera instance	Clock signal name	Clock direction	RCC clock line	Max. frequency	Description
ck_icn_m_dcmipp	DCMIPP	aclk clk_axi_csi	Input	ck_icn_ display	400 MHz	DCMIPP master port interconnect clock
ck_icn_p_dcmipp	DCMIPP	pclk	Input	ck_icn_apb4	150 MHz	DCMIPP peripheral port clock
ck_ker_dcmipp	DCMIPP CSI (host)	clk_proc clk_proc_csi	Input	ck_ker_ dcmipp	333 MHz	Processing clock for Camera subsystem
ck_icn_p_csi	CSI (host)	pclk	Input	ck_icn_apb4	150 MHz	CSI peripheral port clock

Table 154. Camera subsystem external clocks (continued)

Subsystem clock signal name	Camera instance	Clock signal name	Clock direction	RCC clock line	Max. frequency	Description
ck_ker_csiphy	CSI (phy)	i_cfg_clk	Input	ck_ker_ csiphy	27 MHz	-
ck_ker_csitxesc	CSI (phy)	i_TxClkEsc	Input	ck_ker_ csitxesc	20 MHz	-
ck_icn_p_dcmipssi	DCMI/PSSI	hclk_cci	Input	ck_icn_ls _mcu	150 MHz	DCMI and PSSI peripheral port clock
pixck_pdck_cci	DCMI/PSSI DCMIPP	pixck_pdck_cci clk_pix	Input	NA	80 MHz	External DCMI_PIXCLK clock lane
cn_rx_dphy	CSI (phy)	io_Cn	Input/output	NA	1250 MHz	External CSI_CK_N clock lane (2500 Mbps/lane)
cp_rx_dphy	CSI (phy)	io_Cp	Input/output	NA	1250 MHz	External CSI_CK_P clock lane (2500 Mbps/lane)

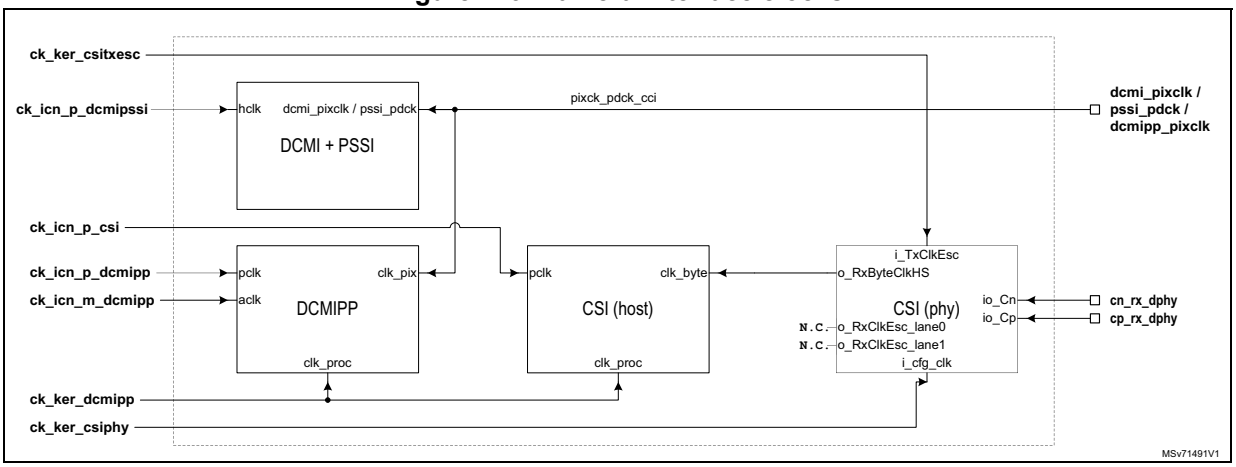
The table below presents the list of the camera subsystem internal clocks.

Table 155. Camera subsystem internal clocks

Functional block	Clock signal name	Clock direction	Max frequency	Description
CSI (phy)	o_RxByteClkHS	Output	312.5 MHz	-

Clocking schematic

Figure 116. Camera interface clocks



20.4.19 Display subsystem clocking

The LTDC uses a clock for its internal pixel processing, which is used to drive its output parallel interface. It also needs clock for its bus master and its bus slave interface.

The table below presents the list of display subsystem external clocks.

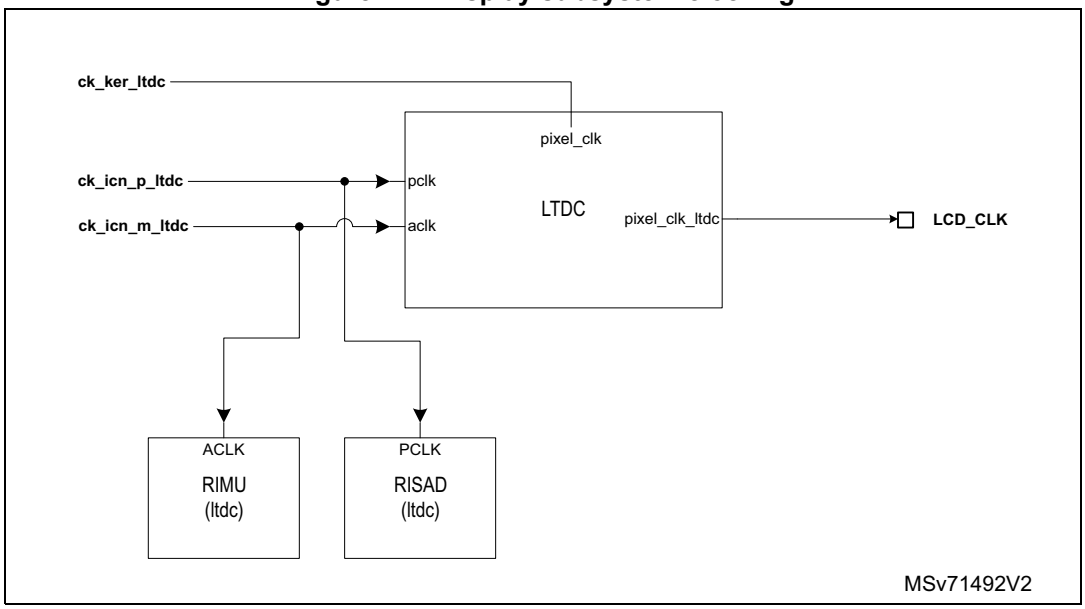
Table 156. Display subsystem external clocks

Subsystem clock signal name	Functional block	Clock signal name	Clock direction	RCC clock line	Max. frequency	Description
ck_icn_m_ltdc	LTDC LTDC (rimu)	aclk ACLK	Input	ck_icn_display	400 MHz	Master port clock
ck_icn_p_ltdc	LTDC LTDC (risad)	pclk PCLK	Input	ck_icn_apb4	150 MHz	Peripheral clock
ck_ker_ltdc	LTDC	pixel_clk	Input	ck_ker_ltdc	314 MHz	Pixel clock Pixel clock LaneByte clock
LCD_CLK	LTDC	pixel_clk_ltdc	Output	NA	150 MHz	External pixel interface clock

Clocking schematic

The clocking tree for the display subsystem is described in the figure below.

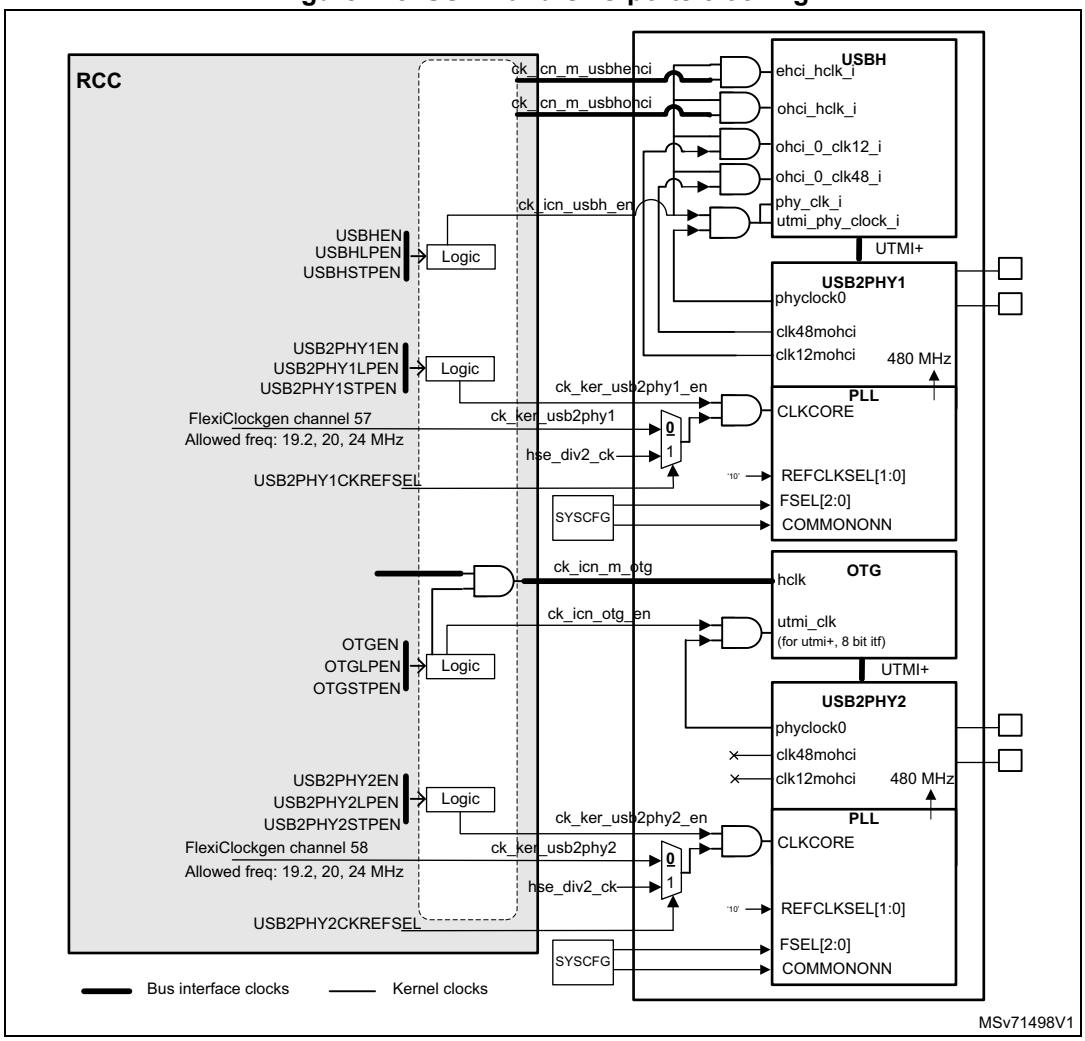
Figure 117. Display subsystem clocking



20.4.20 USB clocking

USB2 Host-only (USBH) and OTG clocking schematic

Figure 118. USBH and OTG ports clocking



USB2 Host-only high speed port

The USB2 Host-only controller (USBH) receives AHB clocks from the RCC, and processes clocks from USB2PHY1.

Table 157. USBH port clocks

Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
USBH	ehci_hclk_i	Input	ck_icn_m_usbhehci	300 MHz	EHCI AHB clock
USBH	ohci_hclk_i	Input	ck_icn_m_usbhohci	300 MHz	OHCI AHB clock
USBH	ohci_0_clk48_i	Input	-	48 MHz	48 MHz reference clock for EHCI controller

Table 157. USBH port clocks (continued)

Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
USBH	ohci_0_clk12_i	Input	-	12 MHz	12 MHz reference clock for OHCI controller
USB2PHY1	CLKCORE	Input	ck_ker_usb2phy1	24 MHz	One of the possible sources of USB2PHY1 reference clock
USB2PHY1	CLKCORE	Input	-	19.2, 20 or 24 MHz	Clock from HSE oscillator: one of the possible sources of USB2PHY1 reference clock

Details of the USB2 Host-only high speed port clocking

The USB2PHY1 PHY receives its reference clock from the RCC or directly from the HSE oscillator (hse_div2_ck clock). This is controlled by the bit USB2PHY1CKREFSEL in RCC_USB2PHY1CFGR register. If the USB2PHY1 reference clock is received from the RCC, it can be sourced from the HSE or re-generated by one of the PLLs and dividers located in the RCC.

The frequency of the USB2PHY1 reference clock can be selected among 19.2, 20 or 24 MHz (50 MHz is reserved for test purpose). This selection is controlled by theUSB2PHY1SEL[2:0] bit-field in SYSCFG_USB2PHY1CR register. These bits must be set to the desired value before USB2PHY1 reset is released.

The clocks provided by USB2PHY1 to the USB2 Host Controller are divisions of a 480 MHz clock that is generated by a PLL inside the PHY (this is a fixed frequency PLL which cannot be programmed).

The USB2PHY1 PLL can be powered down during Suspend and Sleep modes. This is controlled by the USB2PHY1CMN bit in SYSCFG_USB2PHY1CR register. If the OHCI controller is used (FS and LS mode of operation), the USB2PHY1 PLL clock must remain ON during Suspend and Sleep modes to be able to access the OHCI registers.

The clocks provided to the USB2 Host-only port are controlled by enable bits in RCC_USBHCFGR and RCC_USB2PHY1CFGR registers.

USB2 OTG high speed port

The USB2 OTG controller (OTG) receives an AHB clock from the RCC and a processing clock from USB2PHY2.

Table 158. USB2 OTG high speed clocks from RCC and oscillator

Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
OTG	hclk	Input	ck_icn_m_uotg	300 MHz	AHB clock
OTG	utmi_clk	Input	-	60 MHz	Processing clock

Table 158. USB2 OTG high speed clocks from RCC and oscillator (continued)

Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
USB2PHY2	CLKCORE	Input	ck_ker_usb2phy2	24 MHz	One of the possible sources of USB2PHY1 reference clock
USB2PHY2	CLKCORE	Input	-	19.2, 20 or 24 MHz	Clock from HSE oscillator: one of the possible sources of USB2PHY1 reference clock
USB2PHY2	phyclock0	Output	-	60 MHz	UTMI 60 MHz clock (data bus width is 8-bit)

Details of the USB2 Dual Role high speed port clocking

The USB2PHY2 receives its reference clock from the RCC or directly from the HSE oscillator (hse_div2_ck clock). This is controlled by the bit USB2PHY2CKREFSEL in RCC_USB2PHY2CFGR register. If the USB2PHY2 reference clock is received from the RCC, it can be sourced from the HSE or re-generated by one of the PLLs and dividers located in the RCC.

The frequency of the USB2PHY2 reference clock can be selected among 19.2, 20 or 24 MHz (50 MHz is reserved for test purpose). This selection is controlled by the USB2PHY2FSEL[2:0] bit-field in SYSCFG_USB2PHY2CR register. These bits must be set to the desired value before USB2PHY2 reset is released.

The 60 MHz processing clock provided by USB2PHY2 to OTG is obtained by division of the 480 MHz clock which is generated by a PLL inside the PHY (this is a fixed frequency PLL which cannot be programmed).

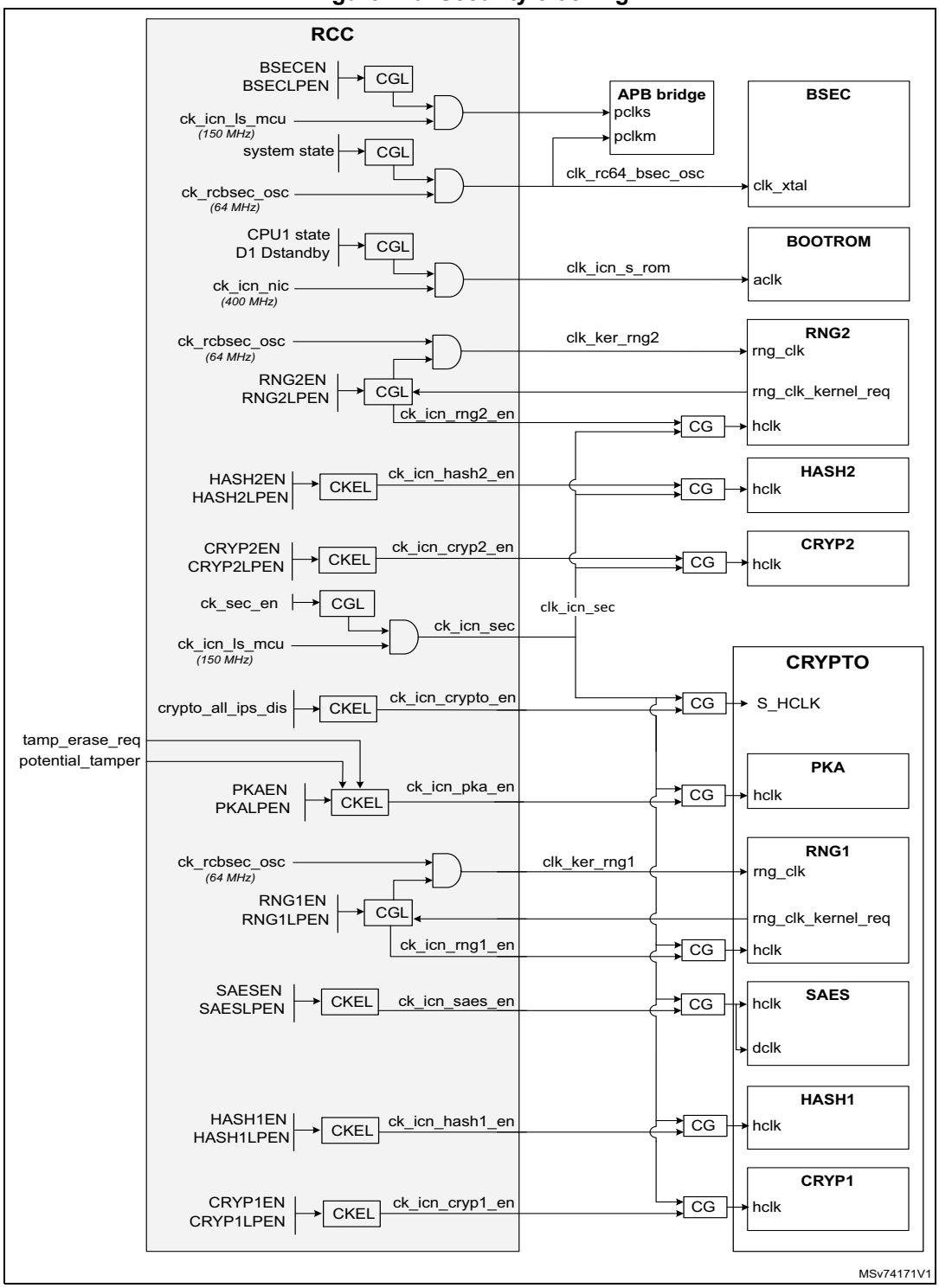
The USB2PHY2 PLL can be powered down during USB Suspend and Sleep modes. This is controlled by the USB2PHY2CMN bit in SYSCFG_USB2PHY2CR register.

The clocks provided by the RCC are controlled by enable bits in RCC_OTGCFGR and RCC_USB2PHY2CFGR registers.

20.4.21 Security clocking

Security interface - Clocking schematic

Figure 119. Security clocking



Note: The peripheral clocks of the following cryptographic peripherals: PKA, RNG1, SAES, HASH1, CRYP1, must all be disabled before issuing a reset request to any of them.

The software should ensure that no cryptographic peripheral is busy (refer to the respective BUSY bit flag of each peripheral (PKA, RNG1, SAES, HASH1, CRYP1) before disabling the peripheral clock.

Table 159. Security clocks from RCC

Subsystem clock signal name	Functional block	Clock signal name	Clock direction	RCC clock line	Max frequency	Description
ck_icn_s_crypto	CRYPTO	clk_icn_sec	Input	ck_icn_sec	150 MHz	AHB clock
ck_icn_p_cryp2	CRYP2		Input			
ck_icn_p_hash2	HASH2		Input			
ck_icn_p_rng2	RNG2		Input			
clk_rc64_rng2		clk_ker_rng2	Input	ck_ker_rng2	64 MHz	kernel clock
clk_rc64_rng1	RNG1	clk_ker_rng1	Input	ck_ker_rng1	64 MHz	kernel clock
clk_icn_s_rom	BOOTROM	aclk	Input	ck_icn_s_bootrom	400 MHz	AXI clock
clk_icn_p_bsec	BSEC	pclks	Input	ck_icn_p_bsec	150 MHz	APB slv clock
clk_rc64_bsec_osc		pclkm	Input	ck_rcbsec	64 MHz	APB mst clock
clk_rc64_bsec_osc		clk_xtal	Input	ck_rcbsec	64 MHz	kernel clock

20.5 RCC interrupts

The RCC provides two independent interrupt interfaces, and interrupt lines for HSI and MSI frequency monitoring:

- an interrupt interface dedicated to the CPU1
- an interrupt interface dedicated to the CPU2

Each interrupt interface provides two interrupt lines:

- a general interrupt line (rcc_cpu1_it or rcc_cpu2_it) that provides events when the PLLs are ready, or when the oscillators are ready
- an interrupt line dedicated to the wake-up of the CPUx when they are in CStop (rcc_cpu1_wkup_it or rcc_cpu2_wkup_it)

All interrupt lines are active high and level triggered.

rcc_cpu1_it and rcc_cpu1_wkup_it interrupts are enabled in RCC_C1CIESETR.

rcc_cpu2_it and rcc_cpu2_wkup_it interrupts are enabled in RCC_C2CIESETR.

rcc_cpu1_it (or rcc_cpu2_it) line is activated when an event flag is set to 1, and its corresponding enable bit is set to 1. In order to release the rcc_cpu1_it (or rcc_cpu2_it) line, all the event flags which have been enabled must be cleared.

rcc_cpu1_wkup_it (or rcc_cpu2_wkup_it) line is activated when the RCC has to exit the CPU1 (or CPU2) from CStop, and if WKUPIE = 1. In order to release the rcc_cpu1_wkup_it (or rcc_cpu2_wkup_it) line, the CPU1 (or CPU2) must write 1 to WKUPF in the corresponding register.

The interrupt flags can be checked and cleared in RCC_C1CIFCLRR or RCC_C2CIFCLRR.

The event flags goes to 1 on rising edge of the corresponding ready bit (xxxRDY).

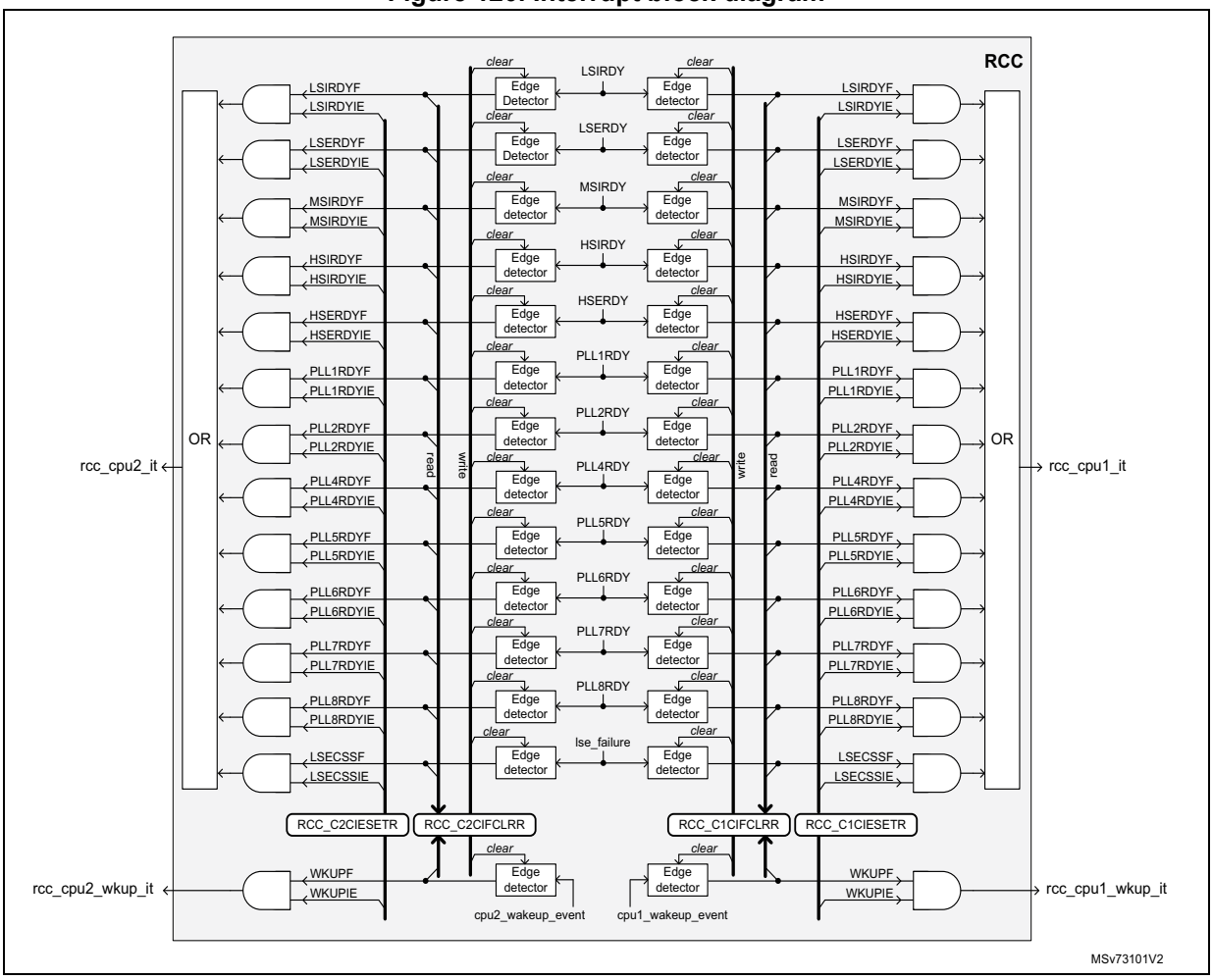
rcc_hsifmon_it interrupt line is used to provide an event to wake up the system when the HSI clock frequency deviates from its defined window while rcc_cpu1_hsifmon_it and rcc_cpu2_hsifmon_it interrupts are used to notify each related CPU.

rcc_hsifmon_it, rcc_cpu1_hsifmon_it, and rcc_cpu2_hsifmon_it interrupts are controlled via RCC_HSIFMONCR.

rcc_msifmon_it interrupt line is used to provide an event to wake up the system when the MSI clock frequency deviates from its defined window while rcc_cpu1_msifmon_it and rcc_cpu2_msifmon_it interrupts are used to notify each related CPU.

rcc_msifmon_it, rcc_cpu1_msifmon_it, and rcc_cpu2_msifmon_it interrupts are controlled via RCC_MSIFMONCR.

Figure 120. Interrupt block diagram



See *Section 20.6.1* for information on CSS for LSE.

Note: The interrupt flags are relevant even if the corresponding interrupt enable bit is not set.

The table below summarizes the interrupt sources, and the way to control them.

Table 160. Interrupt sources and control

Interrupt event flag	Description	Interrupt enable	Action to clear the interrupt(1)	Interrupt lines
LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYF to 1.	rcc_cpu1_it, rcc_cpu2_it
LSERDYF	LSE ready	LSERDYIE	Set LSERDYF to 1.	
HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYF to 1.	
HSERDYF	HSE ready	HSERDYIE	Set HSERDYF to 1.	
MSIRDYF	MSI ready	MSIRDYIE	Set MSIRDYF to 1.	
PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYF to 1.	
PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYF to 1.	
PLL4RDYF	PLL4 ready	PLL4RDYIE	Set PLL4RDYF to 1.	
PLL5RDYF	PLL5 ready	PLL5RDYIE	Set PLL5RDYF to 1.	
PLL6RDYF	PLL6 ready	PLL6RDYIE	Set PLL6RDYF to 1.	
PLL7RDYF	PLL7 ready	PLL7RDYIE	Set PLL7RDYF to 1.	
PLL8RDYF	PLL8 ready	PLL8RDYIE	Set PLL8RDYF to 1.	
LSECSSF	LSE CSS failure	LSECSSIE (2)	Set LSECSSF to 1.	
WKUPF	Wake-up from CStop event	WKUPIE	Set WKUPF to 1.	rcc_cpu1_wkup_it, rcc_cpu2_wkup_it
HSIMONF	HSI frequency monitor flag(3)	HSIMONIE(4)	Set HSIMONF to 1.	rcc_hsifmon_it, rcc_cpu1_hsifmon_it, rcc_cpu2_hsifmon_it
MSIMONF	MSI frequency monitor flag(3)	MSIMONIE(5)	Set MSIMONF to 1.	rcc_msifmon_it, rcc_cpu1_msifmon_it, rcc_cpu2_msifmon_it

1. In RCC_CxCIFCLRR registers.
2. The security system feature must also be enabled (LSECSSON = 1) in order to generate interrupts.
3. Refer to *Section 20.6.3* for details.
4. In RCC_HSIFMONCR register.
5. n RCC_MSIFMONCR register.

20.6 RCC application information

20.6.1 Clock frequency calculation

The flexclkgen engine offers the capability to calculate the frequency of any internal clock via an internal module called clock frequency calculator. For any clock frequency calculation (ckin), the clock frequency calculator requires a reference clock (ckref).

First step - Select the input clock as follows:

1. Set FCALCRSTN in RCC_FCALCOBS1CFGR to enable the clock frequency calculator.

2. Set CKINTSEL (or CKEXTSEL if FCALCCKEXTSEL = 1) in RCC_FCALCOBS0CFGR to select the clock frequency calculator input clock (ckin).
3. Set FCALCCKEN in RCC_FCALCOBS0CFGR to enable the selected input clock at the input of clock frequency calculator.

2nd step - Select the reference clock as follows:

1. Set FCALCREFCKSEL in RCC_FCALCREFCFGR to select the reference clock (ckref).

3rd step - Run the clock frequency calculation as follows:

1. Set FCALCTYP = 0x0C in RCC_FCALCCR2 to select the frequency value type.
2. Select the application mode with FCALCMD in RCC_FCALCCR2.
3. For optimal calculation, loop on the clock frequency time window value with FCALCTWC in RCC_FCALCCR2, from its biggest value (0xF) to its minimum value (0x1) as described below:
   a) Set FCALCTWC biggest value.
   b) Set FCALCRUN in RCC_FCALCCR1.
   c) Poll RCC_FCALCSR until the calculation is valid (FCALCSTS = 1 and FVAL[15:0] not equal to 0)
   d) Reset FCALCRUN in RCC_FCALCCR1.
   e) Check FVAL[16]. If FVAL[16] = 0, exit the loop (FVAL[15:0] is the optimal calculated value). Otherwise continue.
   f) Reset FCALCRSTN in RCC_FCALCOBS1CFGR.
   g) Poll RCC_FCALCSR until it is cleared.
   h) Set FCALCRSTN in RCC_FCALCOBS1CFGR before trying next FCALCTWC value.

Last step - Calculate F_{ckin} clock frequency as follows:

$$TWVAL = ((2^{(FCALCTWC+1)}) + 5) \times CKREFPER$$

with FCALCTWC > 0, and CKREFPER = reference clock period

$$F_{ckin} = \frac{1}{TWVAL / (FVAL - 16)}$$

with F_{ckin} = frequency input clock, TWVAL = clock frequency time window value, and FVAL[15:0] = decimal value

20.6.2 Clock calibration

Frequencies of most of clock source generators can be measured by means of the TIMx input capture, or the clock frequency calculator module.

Calibrating HSI or MSI with the LSE and TIMx

The primary purpose of having the LSE connected to a TIMx input capture is to be able to precisely measure the HSI or MSI. This requires to have the HSI or MSI used as interconnect clock source, via channel 0 the flexclkgen, either directly or via a PLL.
The number of interconnect clock counts between consecutive edges of the LSE provides a calculation of the internal clock period. Taking advantage of the LSE crystal high precision (typically a few tens of ppm), the user can determine the internal clock frequency with the same resolution, and can trim the source to compensate for manufacturing-process, temperature, and/or voltage-related frequency deviations.

The basic concept consists in providing a relative calculation (HSI/LSE ratio): the precision is tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the calculation is.

The HSI and MSI have dedicated user-accessible calibration bits in RCC_HSICFGR and RCC_MSICFGR. When HSI or MSI are used via the PLLx, the interconnect clock can be fine-tuned using the calibration fractional divider.

Calibrating HSI or MSI with the clock frequency calculator

Refer to *Section 20.6.1: Clock frequency calculation*.

20.6.3 Clock monitoring

Monitoring HSI or MSI with LSE

The purpose is to assist the software calibration procedure when the HSI or MSI frequency drifts out of a predefined range because of environmental variation (such as temperature).

This monitoring can be enabled in all system modes except Standby and V_{BAT}, when HSI or MSI is used. When enabled, the number of HSI/MSI kernel clock counts between consecutive edges of the LSE signal provides a calculation of the internal clock period which is checked according to a defined threshold. If the internal clock period is above this threshold, a CPU interrupt (rcc_cpu1_hsifmon_it/rcc_cpu1_msifmon_it or rcc_cpu2_hsifmon_it/rcc_cpu2_msifmon_it), or a wake-up event (rcc_hsifmon_it/rcc_msifmon_it) is triggered to allow the application to trim the HSI/MSI source accordingly.

HSI monitoring control is based on RCC_HSIFMONCR and RCC_HSIFVALR (respectively RCC_MSIFMONCR and RCC_MSIFVALR for MSI). The following steps can be used to enable HSI frequency monitoring (same procedure applies for MSI):

1. Enable LSE signal:
   a) Write 1 to DBP in PWR_BDCR.
   b) Write 1 to in LSEON in RCC_BDCR.
   c) Wait for LSERDY = 1 in RCC_BDCR.
   d) Write 0 to DBP in PWR_BDCR.
2. Setup HSI clock period monitoring:
   a) Write HSIREF in RCC_HSIFMONCR to define the reference period if the default value not suitable (default value = 64 MHz/32.767 kHz = 1953).
   b) Write HSIDEV in RCC_HSIFMONCR to define the threshold if the default value is not suitable (default value = 31, around 1.6% of the reference value).
   c) Write 1 to HSIMONIE in RCC_HSIFMONCR to enable interrupt for application.

d) Write 1 in HSIMONEN in RCC_HSIFMONCR to start monitoring.

The interrupt occurs when the below condition is met:

$$HSIREF + HSIDEV < HSIVAL < HSIREF - HSIDEV$$

Note: MSIREF default value is 488 and MSIDEV default value is 15

20.7 RCC RIF function

To be compliant with the product security definition, the RCC implements the following security/isolation requirements which apply to any defined security domain (see *Section 5.4: Resource isolation framework (RIF)*):

- filtering of secure/non-secure accesses
- filtering of privileged/unprivileged accesses
- filtering of hardware execution compartments (CIDs) such as
  - up to eight CIDs including trusted domain compartment ID (TDCID)
  - dynamic compartment ID (semaphore),
  - debug domain compartment ID (DDCID)

There are four possible profiles for any resource:

- single access (only one master can access)
- shared between different masters with semaphore protection
- shared between different masters without semaphore protection (legacy mode)
- no access

20.7.1 RCC RIF configuration

After any application reset or system reset, all RCC resources are accessible until the trusted domain master agent programs the RCC RIF configuration. This configuration is based on the local and external RIF configurations.

The local RIF configuration is managed by the following registers:

- RCC_SECCFGR0/1/2/3 used to set secure/non-secure attribute per local RCC resource
- RCC_PRIVCFGR0/1/2/3 used to set privileged/unprivileged attribute per local RCC resource
- RCC_RxCIDCFGR and RCC_RxSEMCR used to set the CID filtering policy per local RCC resource
- RCC_SRCFGLOCKR0/1/2/3 are used to lock the local RCC resource configuration.

The external RIF configuration is inherited from the RIF peripheral configuration and it applies to the related RCC_PERxCFGR local resource. This external RIF configuration is used to filter any write access to the related RCC_PERxCFGR register.

The RIF peripheral configuration is driven to the RCC through perxrifcfg signals (with PERx as peripheral x).

Table 161 shows all RCC local resource content (first column) and the RIF configuration attached to it (third column).

Secure/non-secure access filtering

In AMBA5 AHB protocol, the secure/non-secure attribute is propagated via the HNONSEC signal. To enable the filtering access based on this attribute, the authorized master agent has to set SECx in RCC_SECCFGRx related to the RCC resource x.

Any register access with a not-allowed secure attribute is denied (RAZ, WI). An illegal secure access event is generated to the IAC (see *Section 5.4: Resource isolation framework (RIF)* for details). There is no bus error generated.

Privileged/unprivileged access filtering

In AMBA5 AHB protocol, the privileged/unprivileged attribute is propagated via the HPROT[1] signal. To enable the filtering access based on this attribute, the authorized master agent has to set PRIVx in RCC_PRIVCFGRx related to the RCC resource x.

Any register access with a not-allowed privileged attribute is denied (RAZ, WI). An illegal privileged access event is generated to the IAC (see *Section 5.4: Resource isolation framework (RIF)* for details). There is no bus error generated.

CID access filtering

In AMBA5 AHB protocol, the CID attribute is propagated via the HAUSER signals. RCC_RxCIDCFGR controls the CID filtering policy for each non-shareable local resources (without semaphore mechanism). RCC_RxCIDCFGR and RCC_RxSEMCR control the CID filtering policy for each local shareable resources (with semaphore mechanism).

When the compartment-based filtering is active (CFEN = 1):

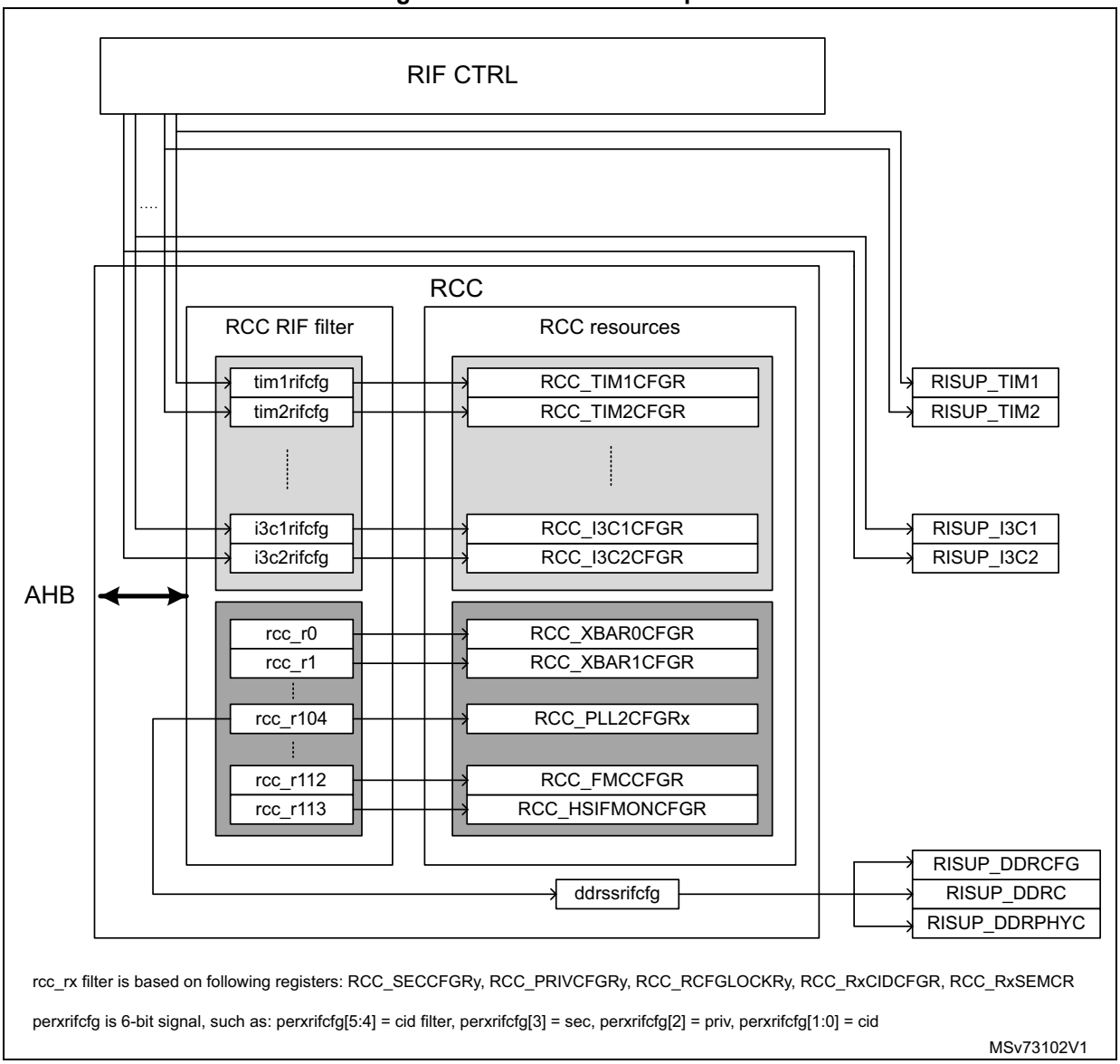
- local non-shareable resources can be individually controlled by a specific CPU. The trusted domain application sets the authorized compartment with SCID in RCC_RxCIDCFGR.
- local shareable resources can be individually controlled by a specific CPU. The trusted domain application sets the compartment values that are authorized to take the local resource, by setting SEMWLCy in RCC_RxCIDCFGR, and sets SEM_EN bit. Each white-listed agent is authorized to take SEM_MUTEX in RCC_RxSEMCR to get the ownership on the resource, until this agent releases the mutex.

A not-allowed CID access to a CID-filtered register is denied (RAZ/WI). An illegal CID access event is generated to the IAC (see *Section 5.4: Resource isolation framework (RIF)* for details). There is no bus error generated.

For debug purpose, the debug domain compartment master, defined by DDCID, can access any resource if secure and privileged attributes are enough.

Note: The RIF configuration (secure and privileged attributes, CID filtering mode) of any RCC local resource can be locked via RCC_RCFGLOCKRx registers.

Figure 121. RCC RIF description



20.8 RCC registers

Some registers can be accessed by all bus masters. Some others can only be accessed by the trusted domain master (see *Section 8: Resource isolation framework security controller (RIFSC)* for details on register accesses).

Note: Any bus transaction initiated by the trusted domain master is secure and privileged.

Table 161 gives the RIF local resource allocation for all RCC registers:

- First column lists all RCC registers defined as a RCC RIF resource number (see second column).
- Second column lists all RCC RIF resources.
- Third column lists RCC registers used to define the RIF configuration of the RCC resource number.

For example, following steps are needed to set the RIF configuration of the RCC RIF resource number 64:

1. A trusted domain agent (with secure, privileged, and TDCID) must program RCC_R64CIDCFGR.
2. A secure agent can set the secure attribute (RCC_SECCFGR2[0] = 1).
3. A privileged agent can set the privileged attribute (RCC_PRIVCFGR2[0] = 1).
4. A secure and privileged agent can lock the RIF configuration (RCC_RCFGLOCKR2[0] = 1).

Table 161. RCC register allocation to RIF resource and access conditions

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R0CIDCFGR	0	Secure, privileged and TDCID/DDCID	RW	Resource 0 CID filtering control
RCC_R0SEMCR		RCC_SECCFGR0[0] RCC_PRIVCFGR0[0] RCC_RCFGLOCKR0[0] RCC_R0CIDCFGR RCC_R0SEMCR		
RCC_XBAR0CFGR				Flexclkgen xbar channel 0
RCC_PREDIV0CFGR				Flexclkgen prediv channel 0
RCC_FINDIV0CFGR				Flexclkgen findiv channel 0
RCC_R1CIDCFGR	1	Secure, Privileged and TDCID/DDCID		Resource 1 CID filtering control
RCC_R1SEMCR		RCC_SECCFGR0[1] RCC_PRIVCFGR0[1] RCC_RCFGLOCKR0[1] RCC_R1CIDCFGR RCC_R1SEMCR		
RCC_XBAR1CFGR				Flexclkgen xbar channel 1
RCC_PREDIV1CFGR				Flexclkgen prediv channel 1
RCC_FINDIV1CFGR				Flexclkgen findiv channel 1
...				

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R63CIDCFGR	63	Secure, privileged and TDCID/DDCID	RW	Resource 63 CID filtering control
RCC_R63SEMCR		RCC_SECCFGR1[31] RCC_PRIVCFGR1[31] RCC_RCFGLOCKR1[31] RCC_R63CIDCFGR RCC_R63SEMCR		
RCC_XBAR63CFGR				Flexclkgen xbar channel 63
RCC_PREDIV63CFGR				Flexclkgen prediv channel 63
RCC_FINDIV63CFGR				Flexclkgen findiv channel 63
RCC_R64CIDCFGR	64	Secure, privileged and TDCID/DDCID		Resource 64 CID filtering control
RCC_R64SEMCR		RCC_SECCFGR2[0] RCC_PRIVCFGR2[0] RCC_RCFGLOCKR2[0] RCC_R64CIDCFGR RCC_R64SEMCR		
RCC_MUXSELCFGR				Flexclkgen reference selection
RCC_PLL4CFGRx				Flexclkgen PLL4 config (x = 1 to 7)
RCC_PLL5CFGRx				Flexclkgen PLL5 config (x = 1 to 7)
RCC_PLL6CFGRx				Flexclkgen PLL6 config (x = 1 to 7)
RCC_PLL7CFGRx				Flexclkgen PLL7 config (x = 1 to 7)
RCC_PLL8CFGRx				Flexclkgen PLL8 config (x = 1 to 7)
RCC_R65CIDCFGR	65	Secure, privileged, and TDCID/DDCID		Resource 65 CID filtering control
RCC_R65SEMCR		RCC_SECCFGR2[1] RCC_PRIVCFGR2[1] RCC_RCFGLOCKR2[1] RCC_R65CIDCFGR RCC_R65SEMCR		
RCC_FCALCOBS0CFGR				Clock frequency calculator and observation 0 clock control
RCC_FCALCOBS1CFGR				Clock frequency calculator and observation 1 clock control
RCC_FCALCREFCFGR				Clock frequency calculator reference clock configuration
RCC_FCALCCR1				Clock frequency calculator control 1
RCC_FCALCCR2				Clock frequency calculator control 2
RCC_R66CIDCFGR	66	Secure, privileged and TDCID/DDCID		Resource 66 CID filtering control
RCC_R66SEMCR		RCC_SECCFGR2[2] RCC_PRIVCFGR2[2] RCC_RCFGLOCKR2[2] RCC_R66CIDCFGR RCC_R66SEMCR		
RCC_GRSTCSETR			SR	System reset set
RCC_C1RSTCSETR				CPU1 reset set
RCC_C2RSTCSETR				CPU2 reset set
RCC_HWRSTSCLRR			CW	Hardware reset status cleared

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R67CIDCFGR	67	Secure, privileged and TDCID/DDCID	RW	Resource 67 CID filtering control
RCC_R67SEMCR		RCC_SECCFGR2[3] RCC_PRIVCFGR2[3] RCC_RCFGLOCKR2[3] RCC_R67CIDCFGR RCC_R67SEMCR		
RCC_CPUBOOTCR				CPUs boot control
RCC_STBYBOOTCR				System Standby exit control
RCC_LEGBOOTCR				Legacy boot control
RCC_PWRLPDLYCR				System Standby exit tempo control
RCC_R68CIDCFGR	68	Secure, privileged and TDCID/DDCID		Resource 68 CID filtering control
RCC_R68SEMCR		RCC_SECCFGR2[4] RCC_PRIVCFGR2[4] RCC_RCFGLOCKR2[4] RCC_R68CIDCFGR RCC_R68SEMCR		
RCC_RDCR				Application reset duration control
RCC_R69CIDCFGR	69	Secure, privileged and TDCID/DDCID		Resource 69 CID filtering control
RCC_R69SEMCR		RCC_SECCFGR2[5] RCC_PRIVCFGR2[5] RCC_RCFGLOCKR2[5] RCC_R69CIDCFGR RCC_R69SEMCR		
RCC_OCENSETR				Oscillator control set
RCC_OCENCLRR				Oscillator control clear
RCC_HSICFGR				HSI oscillator configuration
RCC_MSICFGR				MSI oscillator configuration
RCC_BDCR				V_{SW} domain control
RCC_LSICR				LSI oscillator control
RCC_RTCDIVR				RTC clock divider control
RCC_APB1DIVR				APB1 clock divider control
RCC_APB2DIVR				APB2 clock divider control
RCC_APB3DIVR				APB3 clock divider control
RCC_APB4DIVR				APB4 clock divider control
RCC_APB5DIVR				APB5 clock divider control
RCC_APBDBGDIVR				APBDBG clock divider control
RCC_TIMG1PRER				Timer group1 prescaler control
RCC_TIMG2PRER				Timer group2 prescaler control
RCC_LSMCUDIVR				Low-speed MCU clock divider control

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R70CIDCFGR	70	Secure, privileged and TDCID/DDCID	RW	Resource 70 CID filtering control
RCC_R70SEMCR		RCC_SECCFGR2[6] RCC_PRIVCFGR2[6] RCC_RCFGLOCKR2[6] RCC_R70CIDCFGR RCC_R70SEMCR		
RCC_C1SREQSETR			SR	CPU1 low-power mode control
RCC_C1SREQCLRR			CW	
RCC_C1CIESETR			SR	CPU1 interrupt enable
RCC_C1CIFCLRR			CW	CPU1 interrupt status
RCC_IWDGC1FZSETR			SR	CPU1 IWDG freeze set
RCC_IWDGC1FZCLRR			CR	CPU1 IWDG freeze clear
RCC_IWDGC1CFGSETR			SR	CPU1 IWDG configuration set
RCC_IWDGC1CFGCLRR			CR	CPU1 IWDG configuration cleared
RCC_C1MSRDCR			RW	NRSTC1MS reset delay control
RCC_C1BOOTRSTSSETR			SR	CPU1 boot reset status set
RCC_C1BOOTRSTSCLRR			CW	CPU1 boot reset status cleared
RCC_C1HWRSTSCLRR				CPU1 hardware reset status cleared
RCC_R71CIDCFGR	71	Secure, privileged and TDCID/DDCID	RW	Resource 71 CID filtering control
RCC_R71SEMCR		RCC_SECCFGR2[7] RCC_PRIVCFGR2[7] RCC_RCFGLOCKR2[7] RCC_R71CIDCFGR RCC_R71SEMCR		
RCC_C2CIESETR			SR	CPU2 interrupt enable
RCC_C2CIFCLRR			CW	CPU2 interrupt status
RCC_IWDGC2FZSETR			SR	CPU2 IWDG freeze set
RCC_IWDGC2FZCLRR			CR	CPU2 IWDG freeze clear
RCC_IWDGC2CFGSETR			SR	CPU2 IWDG configuration set
RCC_IWDGC2CFGCLRR			CR	CPU2 IWDG configuration cleared
RCC_C2BOOTRSTSSETR			SR	CPU2 boot reset status set
RCC_C2BOOTRSTSCLRR			CW	CPU2 boot reset status cleared
RCC_C2HWRSTSCLRR				CPU2 hardware reset status cleared
Reserved	72	-		-

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R73CIDCFGR	73	Secure, privileged and TDCID/DDCID	RW	Resource 73 CID filtering control
RCC_R73SEMCR		RCC_SECCFGR2[9] RCC_PRIVCFGR2[9] RCC_RCFGLOCKR2[9] RCC_R73CIDCFGR RCC_R73SEMCR		
RCC_DBGCFGR				Debug control
RCC_STMCFGR				STM control
RCC_ETRCFGR				ETR control
RCC_R74CIDCFGR	74	Secure, privileged and TDCID/DDCID		Resource 74 CID filtering control
RCC_R74SEMCR		RCC_SECCFGR2[10] RCC_PRIVCFGR2[10] RCC_RCFGLOCKR2[10] RCC_R74CIDCFGR RCC_R74SEMCR		
RCC_SYSRAMCFGR				SYSRAM clock control
Reserved	75	-	-	-
RCC_R76CIDCFGR	76	Secure, privileged and TDCID/DDCID	RW	Resource 76 CID filtering control
RCC_R76SEMCR		RCC_SECCFGR2[12] RCC_PRIVCFGR2[12] RCC_RCFGLOCKR2[12] RCC_R76CIDCFGR RCC_R76SEMCR		
RCC_RETRAMCFGR				Retention RAM clock control
RCC_R77CIDCFGR	77	Secure, privileged and TDCID/DDCID		Resource 77 CID filtering control
RCC_R77SEMCR		RCC_SECCFGR2[13] RCC_PRIVCFGR2[13] RCC_RCFGLOCKR2[13] RCC_R77CIDCFGR RCC_R77SEMCR		
RCC_BKPSRAMCFGR				Backup SRAM clock control
RCC_R78CIDCFGR	78	Secure, privileged and TDCID/DDCID		Resource 78 CID filtering control
RCC_R78SEMCR		RCC_SECCFGR2[14] RCC_PRIVCFGR2[14] RCC_RCFGLOCKR2[14] RCC_R78CIDCFGR RCC_R78SEMCR		
RCC_SRAM1CFGR				SRAM1 clock control
Reserved	79	-	-	-
Reserved	80	-	-	-
Reserved	81	-	-	-
Reserved	82	-	-	-

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R83CIDCFGR	83	Secure, privileged and TDCID/DDCID	RW	Resource 83 CID filtering control
RCC_R83SEMCR		RCC_SECCFGR2[19] RCC_PRIVCFGR2[19] RCC_RCFGLOCKR2[19] RCC_R83CIDCFGR RCC_R83SEMCR		
RCC_HPDMA1CFGR				HPDMA1 clock control
RCC_R84CIDCFGR	84	Secure, privileged and TDCID/DDCID		Resource 84 CID filtering control
RCC_R84SEMCR		RCC_SECCFGR2[20] RCC_PRIVCFGR2[20] RCC_RCFGLOCKR2[20] RCC_R84CIDCFGR RCC_R84SEMCR		
RCC_HPDMA2CFGR				HPDMA2 clock control
RCC_R85CIDCFGR	85	Secure, privileged and TDCID/DDCID		Resource 85 CID filtering control
RCC_R85SEMCR		RCC_SECCFGR2[21] RCC_PRIVCFGR2[21] RCC_RCFGLOCKR2[21] RCC_R85CIDCFGR RCC_R85SEMCR		
RCC_HPDMA3CFGR				HPDMA3 clock control
RCC_R87CIDCFGR	87	Secure, privileged and TDCID/DDCID		Resource 87 CID filtering control
RCC_R87SEMCR		RCC_SECCFGR2[23] RCC_PRIVCFGR2[23] RCC_RCFGLOCKR2[23] RCC_R87CIDCFGR RCC_R87SEMCR		
RCC_IPCC1CFGR				IPCC1 clock control
RCC_R88CIDCFGR	88	Secure, privileged and TDCID/DDCID		Resource 88 CID filtering control
RCC_R88SEMCR		RCC_SECCFGR2[24] RCC_PRIVCFGR2[24] RCC_RCFGLOCKR2[24] RCC_R88CIDCFGR RCC_R88SEMCR		
RCC_MSIFMONCR				MSI clock period monitor control
Reserved	89	-	-	-

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R90CIDCFGR	90	Secure, privileged and TDCID/DDCID	RW	Resource 90 CID filtering control
RCC_R90SEMCR		RCC_SECCFGR2[26] RCC_PRIVCFGR2[26] RCC_RCFGLOCKR2[26] RCC_R90CIDCFGR RCC_R90SEMCR		
RCC_GPIOACFGR				GPIOA clock control
RCC_R91CIDCFGR	91	Secure, privileged and TDCID/DDCID		Resource 91 CID filtering control
RCC_R91SEMCR		RCC_SECCFGR2[27] RCC_PRIVCFGR2[27] RCC_RCFGLOCKR2[27] RCC_R91CIDCFGR RCC_R91SEMCR		
RCC_GPIOBCFGR				GPIOB clock control
RCC_R92CIDCFGR	92	Secure, privileged and TDCID/DDCID		Resource 92 CID filtering control
RCC_R92SEMCR		RCC_SECCFGR2[28] RCC_PRIVCFGR2[28] RCC_RCFGLOCKR2[28] RCC_R92CIDCFGR RCC_R92SEMCR		
RCC_GPIOCCFGR				GPIOC clock control
RCC_R93CIDCFGR	93	Secure, privileged and TDCID/DDCID		Resource 93 CID filtering control
RCC_R93SEMCR		RCC_SECCFGR2[29] RCC_PRIVCFGR2[29] RCC_RCFGLOCKR2[29] RCC_R93CIDCFGR RCC_R93SEMCR		
RCC_GPIODCFGR				GPIOD clock control
RCC_R94CIDCFGR	94	Secure, privileged and TDCID/DDCID		Resource 94 CID filtering control
RCC_R94SEMCR		RCC_SECCFGR2[30] RCC_PRIVCFGR2[30] RCC_RCFGLOCKR2[30] RCC_R94CIDCFGR RCC_R94SEMCR		
RCC_GPIOECFGR				GPIOE clock control
RCC_R95CIDCFGR	95	Secure, privileged and TDCID/DDCID		Resource 95 CID filtering control
RCC_R95SEMCR		RCC_SECCFGR2[31] RCC_PRIVCFGR2[31] RCC_RCFGLOCKR2[31] RCC_R95CIDCFGR RCC_R95SEMCR		
RCC_GPIOFCFGR				GPIOF clock control

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R96CIDCFGR	96	Secure, privileged and TDCID/DDCID	RW	Resource 96 CID filtering control
RCC_R96SEMCR		RCC_SECCFGR3[0] RCC_PRIVCFGR3[0] RCC_RCFGLOCKR3[0] RCC_R96CIDCFGR RCC_R96SEMCR		
RCC_GPIOGCFGR				GPIOG clock control
RCC_R97CIDCFGR	97	Secure, privileged and TDCID/DDCID		Resource 97 CID filtering control
RCC_R97SEMCR		RCC_SECCFGR3[1] RCC_PRIVCFGR3[1] RCC_RCFGLOCKR3[1] RCC_R97CIDCFGR RCC_R97SEMCR		
RCC_GPIOHCFGR				GPIOH clock control
RCC_R98CIDCFGR	98	Secure, privileged and TDCID/DDCID		Resource 98 CID filtering control
RCC_R98SEMCR		RCC_SECCFGR3[2] RCC_PRIVCFGR3[2] RCC_RCFGLOCKR3[2] RCC_R98CIDCFGR RCC_R98SEMCR		
RCC_GPIOICFGR				GPIOI clock control
Reserved	99	-	-	-
Reserved	100	-	-	-
RCC_R101CIDCFGR	101	Secure, privileged and TDCID/DDCID	RW	Resource 101 CID filtering control
RCC_R101SEMCR		RCC_SECCFGR3[5] RCC_PRIVCFGR3[5] RCC_RCFGLOCKR3[5] RCC_R101CIDCFGR RCC_R101SEMCR		
RCC_GPIOZCFGR				GPIOZ clock control
RCC_R102CIDCFGR	102	Secure, privileged and TDCID/DDCID		Resource 102 CID filtering control
RCC_R102SEMCR		RCC_SECCFGR3[6] RCC_PRIVCFGR3[6] RCC_RCFGLOCKR3[6] RCC_R102CIDCFGR RCC_R102SEMCR		
RCC_RTCCFGR				RTC clock control
RCC_R103CIDCFGR	103	Secure, privileged and TDCID/DDCID		Resource 103 CID filtering control
RCC_R103SEMCR		RCC_SECCFGR3[7] RCC_PRIVCFGR3[7] RCC_RCFGLOCKR3[7] RCC_R103CIDCFGR RCC_R103SEMCR		
RCC_BSECCFGR				BSEC clock control

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
RCC_R104CIDCFGR	104(2)	Secure, privileged and TDCID/DDCID	RW	Resource 104 CID filtering control
RCC_R104SEMCR		RCC_SECCFGR3[8] RCC_PRIVCFGR3[8] RCC_RCFGLOCKR3[8] RCC_R104CIDCFGR RCC_R104SEMCR		
RCC_PLL2CFGRx				DDR PLL2 control (x=1 to 7)
RCC_DDRCPCFGR				DDR controller AXI ports clock control
RCC_DDRCAPBCFGR				DDR controller APB clock control
RCC_DDRPHYCAPBCFGR				DDR PHY APB clock control
RCC_DDRITFCFGR				DDR interface configuration
RCC_DDRPHYCCFGR				DDRPHY controller configuration
RCC_DDRCFGR				DDR configuration
Reserved	105	-	-	-
RCC_R106CIDCFGR	106	Secure, privileged and TDCID/DDCID	RW	Resource 106 CID filtering control
RCC_R106SEMCR		RCC_SECCFGR3[10] RCC_PRIVCFGR3[10] RCC_RCFGLOCKR3[10] RCC_R106CIDCFGR RCC_R106SEMCR		
RCC_SYSCPU1CFGR				SYSCPU1 clock control
Reserved	107	-	-	-
RCC_R108CIDCFGR	108	Secure, privileged and TDCID/DDCID	RW	Resource 108 CID filtering control
RCC_R108SEMCR		RCC_SECCFGR3[12] RCC_PRIVCFGR3[12] RCC_RCFGLOCKR3[12] RCC_R108CIDCFGR RCC_R108SEMCR		
RCC_MCO1CFGR				MCO1 clock configuration control
RCC_R109CIDCFGR	109	Secure, privileged and TDCID/DDCID		Resource 109 CID filtering control
RCC_R109SEMCR		RCC_SECCFGR3[13] RCC_PRIVCFGR3[13] RCC_RCFGLOCKR3[13] RCC_R109CIDCFGR RCC_R109SEMCR		
RCC_MCO2CFGR				MCO2 clock configuration control
RCC_R110CIDCFGR	110	Secure, privileged and TDCID/DDCID		Resource 110 CID filtering control
RCC_R110SEMCR		RCC_SECCFGR3[14] RCC_PRIVCFGR3[14] RCC_RCFGLOCKR3[14] RCC_R110CIDCFGR RCC_R110SEMCR		
RCC_OSPI1CFGR				OCTOSPI1 clock configuration control

Table 161. RCC register allocation to RIF resource and access conditions (continued)

Register name	RCC RIF resource number	RIF resource configuration	Access type(1)	Description
Reserved	111	-	-	-
RCC_R112CIDCFGR	112	Secure, privileged and TDCID/DDCID	RW	Resource 112 CID filtering control
RCC_R112SEMCR		RCC_SECCFGR3[16] RCC_PRIVCFGR3[16] RCC_RCFGLOCKR3[16] RCC_R112CIDCFGR RCC_R112SEMCR		
RCC_FMCCFGR				FMC clock configuration control
RCC_R113CIDCFGR	113	Secure, privileged and TDCID/DDCID		Resource 113 CID filtering control
RCC_R113SEMCR		RCC_SECCFGR3[17] RCC_PRIVCFGR3[17] RCC_RCFGLOCKR3[17] RCC_R113CIDCFGR RCC_R113SEMCR		
RCC_HSIFMONCR				HSI clock period monitor control
Reserved	114 to 127	-	-	-
RCC_PREDIVSR1	NA	NA	R	Flexclkgen pre divider status flag
RCC_PREDIVSR2				
RCC_FINDIVSR1				Flexclkgen final divider status flag
RCC_FINDIVSR2				
RCC_OCRDYR				Oscillator status clear
RCC_HSIFVALR				HSI clock cycle count
RCC_MSIFVALR				MSI clock cycle count
RCC_FCALCSR				Clock frequency calculator status
RCC_VERR				-
RCC_IDR				-
RCC_SIDR				-

1. RW = read or write, SR = set or read (noted RWS1 in the register description), CR = clear or read (noted rwc1 in the register description, R = read, and W = write.
2. DDR RIF configuration (resource number 104) is exported to the DDR subsystem to filter any DDR programming ports.

All RCC registers can be accessed in word, half-word and byte format, unless explicitly specified differently. All RCC registers are reset by the sys_rstn signal, unless explicitly specified differently.

Note: The burst access mode is not supported; and any opcode fetch transfer generates a bus error.

20.8.1 RCC secure configuration register 0 (RCC_SECCFGR0)

Address offset: 0x000

Reset value: 0x0000 0000

This register is used to set the secure attribute that filters any write access to the RCC local resource number 0 to 31 (see *Table 161* for RCC local resource mapping). Non-secure or unprivileged writes to this register are ignored, and illegal access related events are generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC[31:0]**: Secure attribute reference for local resource number x (x = 31 to 0)

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCKx = 1 in RCC_RCFGLOCKR0.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Secure and non-secure data accesses are granted to local resource number x.

1: Only secure data access is granted to local resource number x.

20.8.2 RCC secure configuration register 1 (RCC_SECCFGR1)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to set the secure attribute value used to filter any write access to the RCC local resources number 32 to 63 (see *Table 161* for RCC local resource mapping). Non-secure or unprivileged writes to this register are ignored, and illegal access related events are generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC[63:32]**: Secure attribute reference for the local resource number x (x = 32 to 63)

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 32) = 1 in RCC_RCFGLOCKR1.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Secure and non-secure data accesses are granted to local resource number x

1: Only secure data access is granted to local resource number x

20.8.3 RCC secure configuration register 2 (RCC_SECCFGR2)

Address offset: 0x008

Reset value: 0x0000 0000

This register is used to set the secure attribute value used to filter any write access to the RCC local resources number 64 to 95 (see *Table 161* for RCC local resource mapping). Non-secure or unprivileged writes to this register are ignored, and illegal access related events are generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC95	SEC94	SEC93	SEC92	SEC91	SEC90	Res.	SEC88	SEC87	SEC86	SEC85	SEC84	SEC83	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC79	SEC78	SEC77	SEC76	SEC75	SEC74	SEC73	Res.	SEC71	SEC70	SEC69	SEC68	SEC67	SEC66	SEC65	SEC64
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 25, 18:16, 8 Reserved, must be kept at reset value.

Bits 31:26, 24:19, 15:9, 7:0 **SECx**: Secure attribute reference for the local resource number x (x = 95 to 90, 88 to 83, 79 to 73, 71 to 64)

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 64) = 1 in RCC_RCFGLOCKR2.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1 it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Secure and non-secure data accesses are granted to local resource number x

1: Only secure data access is granted to local resource number x

20.8.4 RCC secure configuration register 3 (RCC_SECCFGR3)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is used to set the secure attribute value used to filter any write access to the RCC local resources number 96 to 113 (see *Table 161* for RCC local resource

mapping). Non-secure or unprivileged writes to this register are ignored, and illegal access related events are generated.Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC 113	SEC 112
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC 111	SEC 110	SEC 109	SEC 108	Res.	SEC 106	SEC 105	SEC 104	SEC 103	SEC 102	SEC 101	Res.	Res.	SEC 98	SEC 97	SEC 96
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:18, 11, 4:3 Reserved, must be kept at reset value.

Bits 17:12, 10:5, 2:0 **SECx**: Secure attribute reference for the local resource number x (x = 113 to 108, 106 to 101, 98 to 96)

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 96) = 1 in RCC_RCFGLOCKR3.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Non-secure write access is allowed.

1: Secure write access is allowed.

20.8.5 RCC privileged configuration register 0 (RCC_PRIVCFGR0)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to set the privileged attribute value used to filter any write access to the RCC local resource number 0 to 31 (see *Table 161* for RCC local resource mapping). Unprivileged writes to this register are ignored, and illegal access event is generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV[31:0]**: Privileged attribute reference for local resource number x (x = 31 to 0)

If corresponding SECx is set in RCC_SECCFGR0, each bit can only be written by a secure privileged application.

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCKx = 1 in RCC_RCFGLOCKR0.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Privileged and unprivileged data accesses are granted to local resource number x.

1: Only privileged data access is granted to local resource number x.

20.8.6 RCC privileged configuration register 1 (RCC_PRIVCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to set the privileged attribute value used to filter any write access to the RCC local resources number 32 to 63 (see *Table 161* for RCC local resource mapping). Unprivileged writes to this register are ignored, and illegal access event is generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV[63:32]**: Privileged attribute reference for the local resource number x (x = 63 to 32)

If corresponding SEC(x - 32) is set in RCC_SECCFGR1, this bit can only be written by a secure privileged application.

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 32) = 1 in RCC_RCFGLOCKR1.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Privileged and unprivileged data accesses are granted to local resource number x.

1: Only privileged data access is granted to local resource number x.

20.8.7 RCC privileged configuration register 2 (RCC_PRIVCFGR2)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to set the privileged attribute value used to filter any write access to the RCC local resources number 64 to 95 (see *Table 161* for RCC local resource mapping.

Unprivileged writes to this register are ignored, and illegal access event is generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV95	PRIV94	PRIV93	PRIV92	PRIV91	PRIV90	Res.	PRIV88	PRIV87	PRIV86	PRIV85	PRIV84	PRIV83	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV79	PRIV78	PRIV77	PRIV76	PRIV75	PRIV74	PRIV73	Res.	PRIV71	PRIV70	PRIV69	PRIV68	PRIV67	PRIV66	PRIV65	PRIV64
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 25, 18:16, 8 Reserved, must be kept at reset value.

Bits 31:26, 24:19, 15:9, 7:0 **PRIVx**: Privileged attribute reference for the local resource number x (x = 95 to 90, 88 to 83, 79 to 73, 71 to 64)

If corresponding SEC(x - 64) is set in RCC_SECCFGR2, each bit can only be written by a secure privileged application.

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 64) = 1 in RCC_RCFGLOCKR2.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Privileged and unprivileged data accesses are granted to local resource number x.

1: Only privileged data access is granted to local resource number x.

20.8.8 RCC privileged configuration register 3 (RCC_PRIVCFGR3)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to set the privileged attribute value used to filter any write access to the RCC local resources number 96 to 113 (see *Table 161* for RCC local resource mapping). Unprivileged writes to this register are ignored, and illegal access event is generated. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV 113	PRIV 112
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV 111	PRIV 110	PRIV 109	PRIV 108	Res.	PRIV 106	PRIV 105	PRIV 104	PRIV 103	PRIV 102	PRIV 101	Res.	Res.	PRIV 98	PRIV 97	PRIV 96
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:18, 11, 4:3 Reserved, must be kept at reset value.

Bits 17:12, 10:5, 2:0 **PRIVx**: Secure attribute reference for the local resource number x (x = 113 to 108, 106 to 101, 98 to 96)

If corresponding SEC(x - 96) is set in RCC_SECCFGR3, each bit can only be written by a secure privileged application.

If compartment filtering for local resource number x is enabled (resp. disabled) using CFEN in RCC_RxCIDCFGR, only an authorized compartment (resp. any compartment) can write this bit, if access privilege and security are sufficient. Writes are ignored if RLOCK(x - 96) = 1 in RCC_RCFGLOCKR3.

Authorized compartment is defined as SCID if SEM_EN = 0 in RCC_RxCIDCFGR.
If SEM_EN = 1, it is equal to SEMCID if SEM_MUTEX = 1 in RCC_RxSEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

0: Privileged and unprivileged data accesses are granted to local resource number x.

1: Only privileged data access is granted to local resource number x.

20.8.9 RCC resource configuration lock register 0 (RCC_RCFGLOCKR0)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain. Debug domain can write this register if the feature is activated. Otherwise, illegal access events are generated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RLOCK[31:0]**: RCC local resource number x lock (x = 31 to 0)

Each bit is set to lock the RCC local resource x. It is cleared by default, and once set. It cannot be reset until the RCC is reset.

0: SECx in RCC_SECCFGR0 and PRIVx in RCC_PRIVCFGR0 are writable. RCC_RxCIDCFGR is writable.

1: Writes to SECx in RCC_SECCFGR0 and PRIVx in RCC_PRIVCFGR0 are ignored. Writes to RCC_RxCIDCFGR are ignored.

20.8.10 RCC resource configuration lock register 1 (RCC_RCFGLOCKR1)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain. Debug domain can write to this register if the feature is activated. Otherwise, illegal access events are generated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RLOCK[63:32]**: RCC local resource number x lock (x = 63 to 32)

Each bit is set to lock the RCC local resource x. It is cleared by default, and once set, it cannot be reset until RCC is reset.

0: SEC(x - 32) in RCC_SECCFGR1 and PRIV(x - 32) in RCC_PRIVCFGR1 are writable. RCC_RxCIDCFGR is writable.

1: Writes to SEC(x - 32) in RCC_SECCFGR1 and PRIV(x - 32) in RCC_PRIVCFGR1 are ignored. Writes to RCC_RxCIDCFGR are ignored.

20.8.11 RCC resource configuration lock register 2 (RCC_RCFGLOCKR2)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain. Debug domain can write to this register if the feature is activated.Otherwise, illegal access events are generated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK 95	RLOCK 94	RLOCK 93	RLOCK 92	RLOCK 91	RLOCK 90	Res.	RLOCK 88	RLOCK 87	RLOCK 86	RLOCK 85	RLOCK 84	RLOCK 83	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK 79	RLOCK 78	RLOCK 77	RLOCK 76	RLOCK 75	RLOCK 74	RLOCK 73	Res.	RLOCK 71	RLOCK 70	RLOCK 69	RLOCK 68	RLOCK 67	RLOCK 66	RLOCK 65	RLOCK 64
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 25, 18:16, 8 Reserved, must be kept at reset value.

Bits 31:26, 24:19, 15:9, 7:0 **RLOCKx**: RCC local resource number x lock (x = 95 to 90, 88 to 83, 79 to 73, 71 to 64)

Each bit is set to lock the RCC local resource x. It is cleared by default, and once set. It cannot be reset until the RCC is reset.

0: SEC(x - 64) in RCC_SECCFGR2 and PRIV(x - 64) in RCC_PRIVCFGR2 are writable. RCC_RxCIDCFGR is writable.

1: Writes to SEC(x - 64) in RCC_SECCFGR2 and PRIV(x - 64) in RCC_PRIVCFGR2 are ignored. Writes to RCC_RxCIDCFGR are ignored.

20.8.12 RCC resource configuration lock register 3 (RCC_RCFGLOCKR3)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain.Debug domain can write to this register if the feature is activated.Otherwise, illegal access events are generated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLOCK 113	RLOCK 112
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK 111	RLOCK 110	RLOCK 109	RLOCK 108	Res.	RLOCK 106	RLOCK 105	RLOCK 104	RLOCK 103	RLOCK 102	RLOCK 101	Res.	Res.	RLOCK 98	RLOCK 97	RLOCK 96
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:18, 11, 4:3 Reserved, must be kept at reset value.

Bits 17:12, 10:5, 2:0 **RLOCKx**: RCC local resource number x lock (x = 113 to 108, 106 to 101, 98 to 96)
Each bit is set to lock the RCC local resource x. It is cleared by default, and once set. It cannot be reset until the RCC is reset.
0: SEC(x - 96) in RCC_SECCFGR3 and PRIV(x - 96) in RCC_PRIVCFGR3 are writable. RCC_RxCIDCFGR is writable.
1: Writes to SEC(x - 96) in RCC_SECCFGR3 and PRIV(x - 96) in RCC_PRIVCFGR3 are ignored. Writes to RCC_RxCIDCFGR are ignored.

20.8.13 RCC resource x CID configuration register (RCC_RxCIDCFGR)

Address offset: 0x030 + 0x8 * x, (x = 0 to 71, 73 to 79, 83 to 88, 90 to 98, 101 to 106, 108 to 113)

Reset value: 0x0000 0000

This register is used to configure the CID filtering policy for the RCC local resource x (see *Table 161* for RCC local resource mapping). Secure privileged write access only. Write only by trusted domain. Debug domain can write to this register if the feature is activated.

Any write is ignored if the local resource x is locked (RLOCK(x - 32 * z) = 1 in RCC_RCFGLOCKRz, with z = 0 to 3).Otherwise, illegal access events are generated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
									rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SEMWLC[7:0]**: Semaphore white-list for compartment y (y = 7 to 0)
Set and cleared by software.
0: Application running in compartment y cannot get access to local resource x through semaphore SEM_MUTEX.
1: Application running in compartment y can get access to local resource x through semaphore SEM_MUTEX.
Note: This bit is ignored if SEM_EN = 0 or CFEN = 0

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 SCID[2:0]: Static compartment ID
Set by software.
Trusted domain application uses this bit-field to statically define which compartment has access to local resource number x when SEM_EN = 0.
This bit-field is ignored if SEM_EN = 1 or CFEN = 0.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 SEM_EN: Semaphore enable
Set and cleared by software.
0: Access to local resource number x is only granted if the CID of the access is equal to SCID.
1: Access to local resource number x is only granted if SEM_MUTEX = 1 and the CID of the access is equal to SEMCID.
This bit is not taken into account if CFEN = 0. When this bit is cleared, SEM_MUTEX is cleared.

Bit 0 **CFEN**: Static CID filter enable
Set and cleared by software. Additional access controls to local resource x apply, as defined in RCC_PRIVCFGRz and RCC_SECCFGRz with z = 0 to 3.
0: Compartment-based filtering is defined in RCC_RxCIDCFGR, and optionally RCC_RxSEMCR is disabled for local resource x.
1: Compartment-based filtering is defined in RCC_RxCIDCFGR, and optionally RCC_RxSEMCR is enabled for local resource x.
Note: When this bit is cleared, SEM_MUTEX is cleared.

20.8.14 RCC resource x semaphore control register (RCC_RxSEMCR)

Address offset: 0x034 + 0x8 * x**,** (x = 0 to 71, 73 to 79, 83 to 88, 90 to 98, 101 to 106, 108 to 113)

Reset value: 0x0000 0000

This register is used to control the semaphore access for the RCC local resource x.
If compartment filtering is disabled (CFEN = 0 in RCC_RxCIDCFGR), or semaphore is disabled (SEM_EN = 0 in RCC_RxCIDCFGR), any write to this register is ignored. Illegal access event generation is disabled in this case.

Secure-only write access if SEC(x - 32 * z) is set in RCC_SECCFGRz (with z = 0 to 3).
Privileged-only write access if PRIV(x - 32 * z) is set in RCC_PRIVCFGRz (with z = 0 to 3).

If compartment filtering is enabled and semaphore is enabled (CFEN and SEM_EN set in RCC_RxCIDCFGR), only an authorized compartment can write this register. Authorized compartment is defined by SEMCID if SEM_MUTEX = 1. Otherwise, it is defined by the CID white list (SEMWLCy with y = 0 to 7) in RCC_RxCIDCFGR. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_ MUTEX
									r	r	r				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SEMCID[2:0]**: Semaphore compartment ID

Set by hardware. When SEM_EN and CFEN are set in RCC_RxCIDCFGR, this bitfield is updated with the CID value of the latest successful acquisition of RCC_RxSEMCR semaphore (SEM_MUTEX = 1).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX**: Semaphore mutex

Set and cleared by software. White-listed application can set this bit to get exclusive access to local resource x, when SEM_EN = 1 in RCC_RxCIDCFGR (writes ignored otherwise).
When SEM_MUTEX = 0, only applications belonging to the white-listed compartments (defined by SEMWLCy in RCC_RxCIDCFGR) can acquire this semaphore.
When SEM_MUTEX = 1, only applications belonging to the SEMCID compartment can release this semaphore.

0: Reading 0 means that the semaphore is free. Writing 0 means that the semaphore is going to be released.

1: Reading 1 means that the semaphore is taken. Writing 1 means that the semaphore is going to be acquired.

Note: This bit is cleared when CFEN or SEM_EN are cleared in RCC_RxCIDCFGR.

20.8.15 RCC global reset control set register (RCC_GRSTCSETR)

Address offset: 0x400

Reset value: 0x0000 0000

This register is used to generate a system reset. Writing 0 has no effect, reading returns the effective values of the corresponding bits. Writing 1 activates a system reset. Write accesses are defined according to the RCC local resource number 66 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRST
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRST**: System reset
Set by software, cleared by hardware.
0: Writing 0 has no effect. Reading 0 means that the system reset is released.
1: Writing 1 generates a system reset (see *Figure 74*).

20.8.16 RCC CPU1 reset control set register (RCC_C1RSTCSETR)

Address offset: 0x404

Reset value: 0x0000 0000

This register is used to generate the CPU1 reset, and IWDG1/2 kernel resets. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 activates the bit. Write accesses are defined according to the RCC local resource number 66 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1RST
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **C1RST**: CPU1 reset
Set by software, cleared by hardware when the reset command is completed. Resets the CPU1 subsystem, and IWDG1/2 kernel clocks.
0: Writing 0 has no effect. Reading 0 means that the CPU1 reset is released.
1: Writing 1 generates a CPU1 reset. Reading 1 means that the reset is asserted (see *Figure 74*).

20.8.17 RCC CPU2 reset control set register (RCC_C2RSTCSETR)

Address offset: 0x40C

Reset value: 0x0000 0000

This register is used to generate a CPU2 reset, and IWDG3/4 kernel resets. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 activates the reset of the corresponding peripheral. Write accesses are defined according to the RCC local resource number 66 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2RST
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **C2RST**: CPU2 reset

Set by software and cleared by hardware when the reset command is completed. Resets the CPU2 subsystem, and IWDG3/4 kernel clocks.

0: Writing 0 has no effect. Reading 0 means that the CPU2 reset is released.

1: Writing 1 generates a CPU2 reset. Reading 1 means that the reset is asserted (see *Figure 74*).

20.8.18 RCC hardware reset status clear register (RCC_HWRSTSCLRR)

Address offset: 0x410

Reset value: 0x0000 0015

This register is used by the trusted application to check the reset source. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. This register is located in V_{DD} domain, and is reset by por_rstn reset. Write accesses are defined according to the RCC local resource number 66 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RETECCFAILRESTRSTF**: RETRAM ECC failure reset flag during system restoration phase

Cleared by software and set by hardware when a reset due to an ECC failure on RETRAM during the system restoration phase occurs.

0: Writing 0 has no effect. Reading 0 means that no reset due to an ECC failure on RETRAM during the system restoration phase occurred (default after por_rstn reset).

1: Writing 1 clears the RETECCFAILRESTRSTF flag. Reading 1 means that a reset due to an ECC failure on RETRAM during the system restoration phase occurred.

Bit 13 **RETECCFAILCRCRSTF**: RETRAM ECC failure reset flag during the CRC computation phase

Cleared by software and set by hardware when a reset due to an ECC failure on RETRAM during CRC computation phase occurs.

0: Writing 0 has no effect. Reading 0 means that no reset due to an ECC failure on RETRAM during CRC computation phase occurred (default after por_rstn reset).

1: Writing 1 clears the RETECCFAILCRCRSTF flag. Reading 1 means that a reset due to an ECC failure on RETRAM during CRC computation phase occurred.

Bit 12 **RETCRCERRRSTF**: RETRAM CRC error reset flag

Cleared by software and set by hardware when a reset due to a CRC error on RETRAM content occurs.

0: Writing 0 has no effect. Reading 0 means that no reset due to a CRC error on RETRAM content occurred (default after por_rstn reset).

1: Writing 1 clears the RETCRCERRRSTF flag. Reading 1 means that a reset due to a CRC error on RETRAM content occurred.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **IWDG4SYSRSTF**: IWDG4 system reset flag

Cleared by software and set by hardware when an IWDG4 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no IWDG4 system reset occurred (default after por_rstn reset).

1: Writing 1 clears the IWDG4SYSRSTF flag. Reading 1 means that an IWDG4 system reset occurred.

Bit 9 **IWDG3SYSRSTF**: IWDG3 system reset flag

Cleared by software and set by hardware when an IWDG3 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no IWDG3 system reset occurred (default after por_rstn reset).

1: Writing 1 clears the IWDG3SYSRSTF flag. Reading 1 means that an IWDG3 system reset occurred.

Bit 8 **IWDG2SYSRSTF**: IWDG2 system reset flag

Cleared by software and set by hardware when an IWDG2 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no IWDG2 system reset occurred (default after por_rstn reset).

1: Writing 1 clears the IWDG2SYSRSTF flag. Reading 1 means that an IWDG2 system reset occurred.

Bit 7 **IWDG1SYSRSTF**: IWDG1 system reset flag

Cleared by software and set by hardware when an IWDG1 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no IWDG1 system reset occurred (default after por_rstn reset).

1: Writing 1 clears the IWDG1SYSRSTF flag. Reading 1 means that an IWDG1 system reset occurred.

Bit 6 **SYSC2RSTF**: CPU2 system reset flag

Cleared by software and set by hardware when a CPU2 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no system reset generated by the CPU2 occurred (default after por_rstn reset).

1: Writing 1 clears the SYSC2RSTF flag. Reading 1 means that a system reset generated by the CPU2 occurred.

Bit 5 **SYSC1RSTF**: CPU1 system reset flag

Cleared by software and set by hardware when a CPU1 system reset occurs.

0: Writing 0 has no effect. Reading 0 means that no system reset generated by the CPU1 occurred (default after por_rstn reset).

1: Writing 1 clears the SYSC1RSTF flag. Reading 1 means that a system reset generated by the CPU1occurred.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag

Cleared by software and set by hardware when a reset occurs because V_{DDCORE} is lower than the expected value.

0: Writing 0 has no effect. Reading 0 means that V_{DDCORE} is not the origin of the reset.

1: Writing 1 clears the VCORERSTF flag. Reading 1 means that V_{DDCORE} is the origin of the reset (default after por_rstn reset).

Bit 3 **HCSSRSTF**: HSE CSS reset flag

Cleared by software and set by hardware when a failure is detected on HSE.

0: Writing 0 has no effect. Reading 0 means that no HSE CSS reset occurred (default after por_rstn reset).

1: Writing 1 clears the HCSSRSTF flag. Reading 1 means that a HSE CSS reset occurred.

Bit 2 **PADRSTF**: NRST reset flag

Cleared by software and set by hardware when a PAD reset occurs.

0: Writing 0 has no effect. Reading 0 means that no PAD reset occurred.

1: Writing 1 clears the PADRSTF flag. Reading 1 means that a PAD reset occurred (default after por_rstn reset).

Bit 1 **BORRSTF**: BOR reset flag

Cleared by software and set by hardware when a BOR occurs.

0: Writing 0 has no effect. Reading 0 means that no BOR occurred (default after por_rstn reset).

1: Writing 1 clears the BORRSTF flag. Reading 1 means that a BOR occurred.

Bit 0 **PORRSTF**: POR/PDR reset flag

Cleared by software and set by hardware when a POR/PDR occurs.

0: Writing 0 has no effect. Reading 0 means that no POR/PDR occurred.

1: Writing 1 clears the PORRSTF flag. Reading 1 means that a POR/PDR occurred (default after por_rstn reset).

20.8.19 RCC CPU1 hardware reset status clear register (RCC_C1HWRSTSCLRR)

Address offset: 0x414

Reset value: 0x0000 0003

This register is used by the CPU1 to check the reset source. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. This register is located in V_{DD} domain, and is reset by por_rstn reset. This flag is located in the V_{DDCORE} domain, and is reset by sys_rstn reset. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1RSTF	VCPURSTF
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **C1RSTF**: CPU1 reset flag

Cleared by software and set by hardware when a CPU1 reset occurs.

0: Writing 0 has no effect. Reading 0 means that no CPU1 reset occurred.

1: Writing 1 clears the C1RSTF flag. Reading 1 means that a CPU1 reset occurred (default after por_rstn reset).

Bit 0 **VCPURSTF**: V_{DDCPU} reset flag

Cleared by software and set by hardware when a reset occurs because V_{DDCPU} is lower than the expected value.

0: Writing 0 has no effect. Reading 0 means that V_{DDCPU} is not the origin of the reset.

1: Writing 1 clears the VCPURSTF flag. Reading 1 means that V_{DDCPU} is the origin of the reset (default after por_rstn reset).

20.8.20 RCC CPU2 hardware reset status clear register (RCC_C2HWRSTSCLRR)

Address offset: 0x418

Reset value: 0x0000 0001

This register is used by the CPU2 to check the reset source. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. This register is located in the V_{DD} domain, and is reset by por_rstn reset. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2RSTF
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **C2RSTF**: CPU2 reset flag

Cleared by software and set by hardware when a CPU2 reset occurs.

0: Writing 0 has no effect. Reading 0 means that no CPU2 reset occurred.

1: Writing 1 clears the C2RSTF flag. Reading 1 means that a CPU2 reset occurred (default after por_rstn reset).

20.8.21 RCC CPU1 boot reset status set register (RCC_C1BOOTRSTSSETR)

Address offset: 0x41C

Reset value: 0x0000 0000

This register is used by the CPU1 to set the boot reset source. This register is updated by the boot ROM code after any reset source. Writing 0 to this register has no effect. Writing 1 sets the corresponding bits (see *Section 20.3.13* for details). The register is located in the V_{DDCORE} domain. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	D1STBYRSTF	Res.	STBYC1RSTF	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.
								rs	rs		rs	rs	rs	rs	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	C1RSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCPURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
		rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **D2STBYRSTF**: D2 DStandby reset flag
Set by software. Indicates that the D2 domain has been in DStandby mode.
0: No effect
1: Set the D2STBYRSTF bit

Bit 22 **D1STBYRSTF**: D1 DStandby reset flag
Set by software. Indicates that the D1 domain has been in DStandby mode.
0: No effect
1: Set the D1STBYRSTF bit.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **STBYC1RSTF**: CPU1 system Standby reset flag
Set by software. Indicates that the system has been in Standby mode for CPU1.
0: No effect
1: Set the STBYC1RSTF bit.

Bit 19 **RETECCFAILRESTRSTF**: RETRAM ECC failure reset flag during system restoration phase
Set by software. Indicates that a reset due to an ECC failure on RETRAM during the system restoration phase occurred.
0: No effect
1: Set the RETECCFAILRESTRSTF bit.

Bit 18 **RETECCFAILCRCRSTF**: RETRAM ECC failure reset flag during the CRC computation phase
Set by software. Indicates that a reset due to an ECC failure on RETRAM during CRC computation phase occurred.
0: No effect
1: Set the RETECCFAILCRCRSTF bit.

Bit 17 **RETCRCERRRSTF**: RETRAM CRC error reset flag
Set by software. Indicates that a reset due to a CRC error on RETRAM content occurred.
0: No effect
1: Set the RETCRCERRRSTF bit.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **C1RSTF**: CPU1 reset flag
Set by software. Indicates that a CPU1 reset occurred.
0: No effect
1: Set the C1RSTF bit.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **IWDG4SYSRSTF**: IWDG4 system reset flag
Set by software.Indicates that a system reset occurred due to an IWDG4 timeout.
0: No effect
1: Set the IWDG4SYSRSTF bit.

Bit 10 **IWDG3SYSRSTF**: IWDG3 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG3 timeout.
0: No effect
1: Set the IWDG3SYSRSTF bit.

Bit 9 **IWDG2SYSRSTF**: IWDG2 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG2 timeout.
0: No effect
1: Set the IWDG2SYSRSTF bit.

Bit 8 **IWDG1SYSRSTF**: IWDG1 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG1 timeout.
0: No effect
1: Set the IWDG1SYSRSTF bit.

Bit 7 **SYSC2RSTF**: CPU2 system reset flag
Set by software. Indicates that a system reset generated by the CPU2 occurred.
0: No effect
1: Set the SYSC2RSTF bit.

Bit 6 **SYSC1RSTF**: CPU1 system reset flag
Set by software. Indicates that a system reset generated by the CPU1 occurred.
0: No effect
1: Set the SYSC1RSTF bit.

Bit 5 **VCPURSTF**: V_{DDCPU} reset flag
Set by software. Indicates that a reset occurred because V_{DDCPU} is lower than the expected value.
0: No effect
1: Set the VCPURSTF bit.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Set by software. Indicates that a reset occurred because V_{DDCORE} is lower than the expected value.
0: No effect
1: Set the VCORERSTF bit.

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Set by software. Indicates that a reset occurred due to a failure detection on HSE.
0: No effect
1: Set the HCSSRSTF bit.

Bit 2 **PADRSTF**: NRST reset flag
Set by software. Indicates that a pad reset occurred.
0: No effect
1: Set the PADRSTF bit.

Bit 1 **BORRSTF**: BOR reset flag
Set by software. Indicates that a BOR reset occurred.
0: No effect
1: Set the BORRSTF bit.

Bit 0 **PORRSTF**: POR/PDR reset flag
Set by software. Indicates that a POR/PDR reset occurred.
0: No effect
1: Set the PORRSTF bit.

20.8.22 RCC CPU1 boot reset status clear register (RCC_C1BOOTRSTSCLRR)

Address offset: 0x420

Reset value: 0x0000 0000

This register is used by the CPU1 to check the boot reset source. This register is updated by the boot ROM code after any reset source. Writing 0 to this register has no effect. Writing 1

clears the corresponding bits (see *Section 20.3.13* for details). The register is located in the V_{DDCORE} domain. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	D1STBYRSTF	Res.	STBYC1RSTF	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.
								rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	C1RSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCPURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **D2STBYRSTF**: D2 DStandby reset flag
Cleared by software. Indicates that the D2 domain has been in DStandby mode.
0: No effect
1: Clear the D2STBYRSTF bit.

Bit 22 **D1STBYRSTF**: D1 DStandby reset flag
Cleared by software. Indicates that the D1 domain has been in DStandby mode.
0: No effect
1: Clear the D1STBYRSTF bit.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **STBYC1RSTF**: CPU1 system Standby reset flag
Cleared by software. Indicates that the system has been in Standby mode for CPU1.
0: No effect
1: Clear the STBYC1RSTF bit.

Bit 19 **RETECCFAILRESTRSTF**: RETRAM ECC failure reset flag during system restoration phase
Cleared by software. Indicates that a reset due to an ECC failure on RETRAM during the system restoration phase occurred.
0: No effect
1: Clear the RETECCFAILRESTRSTF bit.

Bit 18 **RETECCFAILCRCRSTF**: RETRAM ECC failure reset flag during the CRC computation phase
Cleared by software. Indicates that a reset due to an ECC failure on RETRAM during CRC computation phase occurred.
0: No effect
1: Clear the RETECCFAILCRCRSTF bit.

Bit 17 **RETCRCERRRSTF**: RETRAM CRC error reset flag
Cleared by software. Indicates that a reset due to a CRC error on RETRAM content occurred.
0: No effect
1: Clear the RETCRCERRRSTF bit.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **C1RSTF**: CPU1 reset flag
Cleared by software. Indicates that a CPU1 reset occurred.
0: No effect
1: Clear the C1RSTF bit.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **IWDG4SYSRSTF**: IWDG4 system reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG4 timeout.
0: No effect
1: Clear the IWDG4SYSRSTF bit.

Bit 10 **IWDG3SYSRSTF**: IWDG3 system reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG3 timeout.
0: No effect
1: Clear the IWDG3SYSRSTF bit.

Bit 9 **IWDG2SYSRSTF**: IWDG2 system reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG2 timeout.
0: No effect
1: Clear the IWDG2SYSRSTF bit.

Bit 8 **IWDG1SYSRSTF**: IWDG1 system reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG1 timeout.
0: No effect
1: Clear the IWDG1SYSRSTF bit.

Bit 7 **SYSC2RSTF**: CPU2 system reset flag
Cleared by software. Indicates that a system reset generated by the CPU2 occurred.
0: No effect
1: Clear the SYSC2RSTF bit.

Bit 6 **SYSC1RSTF**: CPU1 system reset flag
Cleared by software. Indicates that a system reset generated by the CPU1 occurred.
0: No effect
1: Clear the SYSC1RSTF bit.

Bit 5 **VCPURSTF**: V_{DDCPU} reset flag
Cleared by software. Indicates that a reset occurred because V_{DDCPU} is lower than the expected value.
0: No effect
1: Clear the VCPURSTF bit.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Cleared by software. Indicates that a reset occurred because V_{DDCORE} is lower than the expected value.
0: No effect
1: Clear the VCORERSTF bit.

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Cleared by software. Indicates that a reset occurred due to a failure detection on HSE.
0: No effect
1: Clear the HCSSRSTF bit.

Bit 2 **PADRSTF**: NRST reset flag
Cleared by software. Indicates that a pad reset occurred.
0: No effect
1: Clear the PADRSTF bit.

Bit 1 **BORRSTF**: BOR reset flag
Cleared by software. Indicates that a BOR reset occurred.
0: No effect
1: Clear the BORRSTF bit.

Bit 0 **PORRSTF**: POR/PDR reset flag
Cleared by software. Indicates that a POR/PDR reset occurred.
0: No effect
1: Clear the PORRSTF bit.

20.8.23 RCC CPU2 boot reset status set register (RCC_C2BOOTRSTSSETR)

Address offset: 0x424

Reset value: 0x0000 0000

This register is used by the CPU2 to set the boot reset source. This register is updated by the CPU2 wake-up firmware after any reset source. Writing 0 to this register has no effect. Writing 1 sets the corresponding bits (see *Section 20.3.13* for details). The register is located in the V_{DDCORE} domain. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details).Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	Res.	STBYC2RSTF	Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.
								rs		rs		rs	rs	rs	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	C2RSTF	Res.	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	rs			rs	rs	rs	rs	rs	rs		rs	rs	rs	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **D2STBYRSTF**: D2 DStandby reset flag
Set by software. Indicates that the D2 domain has been in DStandby mode.
0: No effect
1: Set the D2STBYRSTF bit.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **STBYC2RSTF**: CPU2 system Standby reset flag
Set by software. Indicates that the system has been in Standby mode for CPU2.
0: No effect
1: Set the STBYC2RSTF bit.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RETECCFAILRESTRSTF**: RETRAM ECC failure reset flag during system restoration phase
Set by software. Indicates that a reset due to an ECC failure on RETRAM during the system restoration phase occurred.
0: No effect
1: Set the RETECCFAILRESTRSTF bit.

Bit 18 **RETECCFAILCRCRSTF**: RETRAM ECC failure reset flag during the CRC computation phase
Set by software. Indicates that a reset due to an ECC failure on RETRAM during CRC computation phase occurred.
0: No effect
1: Set the RETECCFAILCRCRSTF bit.

Bit 17 **RETCRCERRRSTF**: RETRAM CRC error reset flag
Set by software. Indicates that a reset due to a CRC error on RETRAM content occurred.
0: No effect
1: Set the RETCRCERRRSTF bit.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **C2RSTF**: CPU2 reset flag
Set by software. Indicates that a CPU2 reset occurred.
0: No effect
1: Set the C2RSTF bit.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **IWDG4SYSRSTF**: IWDG4 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG4 timeout.
0: No effect
1: Set the IWDG4SYSRSTF bit.

Bit 10 **IWDG3SYSRSTF**: IWDG3 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG3 timeout.
0: No effect
1: Set the IWDG3SYSRSTF bit.

Bit 9 **IWDG2SYSRSTF**: IWDG2 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG2 timeout.
0: No effect
1: Set the IWDG2SYSRSTF bit.

Bit 8 **IWDG1SYSRSTF**: IWDG1 system reset flag
Set by software. Indicates that a system reset occurred due to an IWDG1 timeout.
0: No effect
1: Set the IWDG1SYSRSTF bit.

Bit 7 **SYSC2RSTF**: CPU2 system reset flag
Set by software. Indicates that a system reset generated by the CPU2 occurred.
0: No effect
1: Set the SYSC2RSTF bit.

Bit 6 **SYSC1RSTF**: CPU1 system reset flag
Set by software. Indicates that a system reset generated by the CPU1 occurred.
0: No effect
1: Set the SYSC1RSTF bit.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Set by software. Indicates that a reset occurred because V_{DDCORE} is lower than the expected value.
0: No effect
1: Set the VCORERSTF bit.

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Set by software. Indicates that a reset occurred due to a failure detection on HSE.
0: No effect
1: Set the HCSSRSTF bit.

Bit 2 **PADRSTF**: NRST reset flag
Set by software. Indicates that a pad reset occurred.
0: No effect
1: Set the PADRSTF bit.

Bit 1 **BORRSTF**: BOR reset flag
Set by software. Indicates that a BOR reset occurred.
0: No effect
1: Set the BORRSTF bit.

Bit 0 **PORRSTF**: POR/PDR reset flag
Set by software. Indicates that a POR/PDR reset occurred.
0: No effect
1: Set the PORRSTF bit.

20.8.24 RCC CPU2 boot reset status clear register (RCC_C2BOOTRSTSCLRR)

Address offset: 0x428

Reset value: 0x0000 0000

This register is used by the CPU2 to check the boot reset source. This register is updated by the CPU2 wake-up firmware after any reset source. Writing 0 to this register has no effect. Writing 1 clears the corresponding bits (see *Section 20.3.13* for details). The register is located in the V_{DDCORE} domain. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	Res.	STBYC2RSTF	Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.
								rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	C2RSTF	Res.	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **D2STBYRSTF**: D2 DStandby reset flag
Cleared by software. Indicates that the D2 domain has been in DStandby mode.
0: No effect
1: Clear the D2STBYRSTF bit.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **STBYC2RSTF**: CPU2 system Standby reset flag
Cleared by software. Indicates that the system has been in Standby mode for CPU2.
0: No effect
1: Clear the STBYC2RSTF bit.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RETECCFAILRESTRSTF**: RETRAM ECC failure reset flag during system restoration phase
Cleared by software. Indicates that a reset due to an ECC failure on RETRAM during the system restoration phase occurred.
0: No effect
1: Clear the RETECCFAILRESTRSTF bit.

Bit 18 **RETECCFAILCRCRSTF**: RETRAM ECC failure reset flag during the CRC computation phase
Cleared by software. Indicates that a reset due to an ECC failure on RETRAM during CRC computation phase occurred.
0: No effect
1: Clear the RETECCFAILCRCRSTF bit.

Bit 17 **RETCRCERRRSTF**: RETRAM CRC error reset flag
Cleared by software. Indicates that a reset due to a CRC error on RETRAM content occurred.
0: No effect
1: Clear the RETCRCERRRSTF bit.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **C2RSTF**: CPU2 reset flag
Cleared by software. Indicates that a CPU2 reset occurred.
0: No effect
1: Clear the C2RSTF bit.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **IWDG4SYSRSTF**: IWDG4 reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG4 timeout.
0: No effect
1: Clear the IWDG4SYSRSTF bit.

Bit 10 **IWDG3SYSRSTF**: IWDG3 reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG3 timeout.
0: No effect
1: Clear the IWDG3SYSRSTF bit.

Bit 9 **IWDG2SYSRSTF**: IWDG2 reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG2 timeout.
0: No effect
1: Clear the IWDG2SYSRSTF bit.

Bit 8 **IWDG1SYSRSTF**: IWDG1 system reset flag
Cleared by software. Indicates that a system reset occurred due to an IWDG1 timeout.
0: No effect
1: Clear the IWDG1SYSRSTF bit.

Bit 7 **SYSC2RSTF**: CPU2 system reset flag
Cleared by software. Indicates that a system reset generated by the CPU2 occurred.
0: No effect
1: Clear the SYSC2RSTF bit.

Bit 6 **SYSC1RSTF**: CPU1 system reset flag
Cleared by software. Indicates that a system reset generated by the CPU1 occurred.
0: No effect
1: Clear the SYSC1RSTF bit.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Cleared by software. Indicates that a reset occurred because V_{DDCORE} is lower than the expected value.
0: No effect
1: Clear the VCORERSTF bit.

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Cleared by software. Indicates that a reset occurred due to a failure detection on HSE.
0: No effect
1: Clear the HCSSRSTF bit.

Bit 2 **PADRSTF**: NRST reset flag
Cleared by software. Indicates that a pad reset occurred.
0: No effect
1: Clear the PADRSTF bit.

Bit 1 **BORRSTF**: BOR reset flag
Cleared by software. Indicates that a BOR reset occurred.
0: No effect
1: Clear the BORRSTF bit.

Bit 0 **PORRSTF**: POR/PDR reset flag
Cleared by software. Indicates that a POR/PDR reset occurred.
0: No effect
1: Clear the PORRSTF bit.

20.8.25 RCC CPU1 stop request set register (RCC_C1SREQSETR)

Address offset: 0x42C

Reset value: 0x0000 0000

Writing 0 has no effect. Reading returns the values of the bits. Writing 1 sets the corresponding bit to 1. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESLPREQ
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P0
															rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ESLPREQ**: Enhanced CSleep request for CPU1 subsystem

Set by software. Allows the entry in enhanced CSleep for CPU1 sub-ystem, when the complete CPU1 is in WFI.

0: Writing 0 has no effect. Reading 0 means that enhanced CSleep for CPU1 subsystem is not requested.

1: Writing 1 sets the ESLPREQ bit. Reading 1 means that enhanced CSleep for CPU1 subsystem is requested.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **STPREQ_P0**: Stop request for CPU1 processor

Set by software. Indicates if the CPU1 processor allows the entry in Stop mode for CPU1 subsystem, when the CPU1 is in WFI.

0: Writing 0 has no effect. Reading 0 means that the CPU1 processor does not allow the CPU1 subsystem to go to CStop.

1: Writing 1 sets the STPREQ_P0 bit. Reading 1 means that the CPU1 processor allows the CPU1 subsystem to go to CStop.

20.8.26 RCC CPU1 stop request clear register (RCC_C1SREQCLRR)

Address offset: 0x430

Reset value: 0x0000 0000

Writing 0 has no effect. Reading returns the effective values of the bits. Writing 1 clears the corresponding bit to 0. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESLPREQ
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P0
															rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ESLPREQ**: Enhanced CSleep request for CPU1 subsystem

Cleared by software. Allows the entry in enhanced CSleep for CPU1 subsystem, when the complete CPU1 is in WFI.

0: Writing 0 has no effect. Reading 0 means that enhanced CSleep for CPU1 subsystem is not requested.

1: Writing 1 clears the ESLPREQ bit. Reading 1 means that enhanced CSleep for the CPU1 subsystem is requested.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **STPREQ_P0**: Stop request for CPU1 processor

Cleared by software. Indicates if the CPU1 processor allows the entry in Stop mode for CPU1 subsystem, when the CPU1 is in WFI.

0: Writing 0 has no effect. Reading 0 means that the CPU1 processor does not allow the CPU1 subsystem to go to CStop.

1: Writing 1 clears the STPREQ_P0 bit. Reading 1 means that the CPU1 processor allows the CPU1 subsystem to go to CStop.

20.8.27 RCC CPU boot control register (RCC_CPUBOOTCR)

Address offset: 0x434

Reset value: 0x0000 0000

The register is used to control the HOLD_BOOT state of the CPUs after reset (see *HOLD_BOOT CPUx state* for details). Write accesses are defined according to the

RCC local resource number 67 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_CPU1	BOOT_CPU2
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **BOOT_CPU1**: CPU1 boot

Set and cleared by software to set the CPU1 in HOLD_BOOT after a reset of the CPU1. If the CPU1 was in HOLD_BOOT, it causes the CPU1 to boot.

0: The CPU1 is set in HOLD_BOOT when the next CPU1 reset occurs.

1: The CPU1 is not in HOLD_BOOT mode when the next CPU1 reset occurs.

Bit 0 **BOOT_CPU2**: CPU2 to boot

Set and cleared by software to set the CPU2 in HOLD_BOOT after a reset of the CPU2 core. If the CPU2 was in HOLD_BOOT, it causes the CPU2 to boot.

0: The CPU2 is set in HOLD_BOOT when the next CPU2 core reset occurs.

1: The CPU2 is not in HOLD_BOOT mode when the next CPU2 core reset occurs.

20.8.28 RCC standby boot control register (RCC_STBYBOOTCR)

Address offset: 0x438

Reset value: 0x0000 0000

This register is used to control the boot sequence when the system exits Standby (see *Section 20.3.13* for details). This register is reset when a system reset occurs (app_rstn), but not when the circuit exits from Standby. Wait states are inserted in case of successive write accesses to this register. The number of wait states can be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 67 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET_CRCERR_RSTEN	Res.	Res.	CPU1_HW_BEN	CPU2_HW_BEN	Res.	COLD_CPU2	CPU_BEN_SEL	Res.
							rw			r	r		rwo	rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RET_CRCERR_RSTEN**: RETRAM CRC error system reset enable
Set and cleared by software to select whether a RETRAM CRC error triggers a system reset or not.
0: RETRAM CRC error does not trigger a system reset (default after reset).
1: RETRAM CRC error triggers a system reset.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CPU1_HW_BEN**: CPU1 hardware boot enable
Set and cleared by hardware to indicate whether the CPU1 is enabled to restart after the system exits Standby.
0: CPU1 remains in HOLD_BOOT state.
1: CPU1 is allowed to restart.

Bit 4 **CPU2_HW_BEN**: CPU2 hardware boot enable
Set and cleared by hardware to indicate whether the CPU2 is enabled to restart after the system exits Standby.
0: CPU2 remains in HOLD_BOOT state.
1: CPU2 is allowed to restart.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **COLD_CPU2**: CPU2 cold boot
Set by software to indicate that CPU2 started first after POR. This bit can be written only one time after system reset.
0: CPU1 is the cold boot CPU.
1: CPU2 is the cold boot CPU.

Bit 1 **CPU_BEN_SEL**: CPU boot select
Set and cleared by software to select the CPU to restart first after the system exits Standby, in wake-up event mode in case of simultaneous CPU events.
0: CPU1 restarts first (default after reset).
1: CPU2 restarts first.

Bit 0 Reserved, must be kept at reset value.

20.8.29 RCC legacy boot control register (RCC_LEGBOOTCR)

Address offset: 0x43C

Reset value: 0x0000 0000

This register is used to control the first CPUx to boot (see *Section 20.3.13* for details). This register is reset when a system reset (app_rstn) occurs but not when the circuit exits Standby. Wait states are inserted in case of successive write accesses to this register. The number of wait states can be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 67 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEGACY_BEN
															rwo

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LEGACY_BEN**: Enable of the legacy selection mode to choose the first CPU to boot
Set by software to indicate that the CPU1 boots first. This bit can be written only one time after system reset.
0: The legacy selection mode is disabled (default after reset).
1: The legacy selection mode is enabled.

20.8.30 RCC backup domain control register (RCC_BDCR)

Address offset: 0x440

Reset value: 0x0000 0020

This register is used to control LSE functions. Wait states are inserted in case of successive write accesses to this register. The number of wait states may be up to 7 cycles of the RCC clock. After a system reset, this register is write-protected. In order to modify this register, DBP in PWR_BDCR must be set to 1. This register is only reset after a backup domain reset (vsw_rstn, see *Section 20.3.6*). Any other internal or external reset does not have any effect on these bits. This register is located in the V_{SW} domain.
Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VSWRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCCKEN	Res.	Res.	RTCSRC[1:0]	
rw											rw			rwo	rwo
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSD	LSEGFON	LSECSSON	LSEDRV[1:0]		LSEDIGBYP	LSERDY	LSEBYP	LSEON
							r	rw	rw	rw	rw	rw	r	rw	rw

Bit 31 **VSWRST**: V_{SW} domain software reset
Set and cleared by software.
0: Reset not activated (default after backup domain reset)
1: Resets the entire V_{SW} domain.

Bits 30:21 Reserved, must be kept at reset value.

Bit 20 **RTCCKEN**: RTC clock enable
Set and cleared by software.
0: rtc_ck clock disabled (default after backup domain reset)
1: rtc_ck clock enabled

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **RTCSRC[1:0]**: RTC clock source selection
Set by software to select the clock source for the RTC. This field can be written only one time after backup domain reset. When an LSE failure occurs, this field is cleared to 0, and the software can write it again once. This field must be written before LSECSSON is enabled (see *CSS on LSE* for details).
0x0: No clock (default after backup domain reset)
0x1: LSE clock used as RTC clock
0x2: LSI clock used as RTC clock
0x3: HSE clock divided by RTCDIV value used as RTC clock

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **LSECSSD**: LSE CSS failure detection

Set by hardware to indicate when a failure has been detected by the CSS on the external 32 kHz oscillator.

0: No failure detected on 32 kHz oscillator (default after backup domain reset)

1: Failure detected on 32 kHz oscillator

Bit 7 **LSEGFON**: LSE clock glitch filter enable

Set and cleared by software to enable the LSE clock glitch filter. This bit can be written only when LSE is disabled (LSEON = 0).

0: LSE clock glitch filter disabled

1: LSE clock glitch filter enabled

Bit 6 **LSECSSON**: LSE CSS enable

Set by software to enable the CSS on 32 kHz oscillator (see *LSE oscillator* for details on the activation and deactivation sequences). Once this bit is set, it cannot be cleared, except after a LSE failure detection (LSECSSD = 1).

0: CSS on 32 kHz oscillator off (default after backup domain reset)

1: CSS on 32 kHz oscillator on

Bits 5:4 **LSEDRV[1:0]**: LSE oscillator driving capability

Set by software to select the driving capability of the LSE oscillator.

0x0: Lowest drive

0x1: Medium-high drive

0x2: Medium-low drive (default after backup domain reset)

0x3: Highest drive

Caution: The low drive to high drive programming values are not in increasing order of LSEDRV[1:0].

Bit 3 **LSEDIGBYP**: LSE digital bypass

Set and cleared by software to select the analog or digital bypass mode

0: LSE in analog bypass mode (default after backup domain reset)

1: LSE in digital bypass mode

Bit 2 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the LSE is stable. This bit needs 6 cycles of lse_ck to fall down after LSEON has been cleared to 0.

0: LSE oscillator not ready (default after backup domain reset)

1: LSE oscillator ready

Bit 1 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (LSEON = 1) (see *LSE oscillator* for details).

0: LSE oscillator not bypassed (default after backup domain reset)

1: LSE oscillator bypassed

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.

0: LSE oscillator off (default after backup domain reset)

1: LSE oscillator on

20.8.31 RCC reset duration control register (RCC_RDCR)

Address offset: 0x44C

Reset value: 0x0600 0000

This register is used to control the minimum NRST active duration. Wait states are inserted in case of successive write accesses to this register. The number of wait states may be up to 7 cycles of the RCC clock. This register is reset by por_rstn reset, and it is located in the V_{DD} domain. Write accesses are defined according to the RCC local resource number 68 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	EADLY[3:0]				Res.	Res.	Res.	MRD[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **EADLY[3:0]**: External access delays

Set and cleared by software. Time to wait before boot ROM performs any external access (for peripherals like UART, USB, OCTOSPI, FMC, or SDMMC).

0x0: No extra delay added by boot ROM
0x1: 100 microseconds
0x2: 200 microseconds
0x3: 500 microseconds
0x4: 1 milliseconds
0x5: 2 milliseconds
0x6: 5 milliseconds (default after reset)
0x7: 10 milliseconds
0x8: 20 milliseconds
0x9: 50 milliseconds
0xA: 100 milliseconds
0xB: 200 milliseconds
0xC: 500 milliseconds
0xD: 1 second
0xE: 2 seconds
0xF: 5 seconds

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **MRD[4:0]**: Minimum reset duration

Set and cleared by software. This field defines the minimum guaranteed duration of the NRST low pulse. The LSI oscillator is automatically enabled when needed by the RPCTL.

0x0: NRST low-pulse duration is guaranteed by the pulse stretcher of the pin. The RPCTL is bypassed (default after reset).
0x1: The guaranteed NRST low-pulse duration is about 1 ms (1 x 32 lsi_ck cycles).
0x2: The guaranteed NRST low-pulse duration is about 2 ms (2 x 32 lsi_ck cycles).
...
0x1F:The guaranteed NRST low-pulse duration is about 31 ms (31 x 32 lsi_ck cycles).

Bits 15:0 Reserved, must be kept at reset value.

20.8.32 RCC C1MS reset delay control register (RCC_C1MSRDCR)

Address offset: 0x450

Reset value: 0x0000 0000

This register is used to control the minimum NRSTC1MS active duration after any D1 Standby exit. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1MSRST	Res.	Res.	Res.	C1MSRD[4:0]				
							rs				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **C1MSRST**: Trigger low-level pulse on NRSTC1MS pin
Set by software and cleared by hardware.
0: Reading 0 means that the NRSTC1MS level is high (inactive) (default after reset).
1: Writing 1 triggers a low-level pulse on NRSTC1MS according to C1MSRD value.
Reading 1 means that the NRSTC1MS level is low (active).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **C1MSRD[4:0]**: Minimum Standby reset duration
Set and cleared by software. This filed defines the minimum duration of the low-level pulse on NRSTC1MS when C1MSRST is high. The LSI is automatically enabled after any D1 DStandby exit.
0x0: 'No pulse on NRSTC1MS (default after reset)
0x1: The guaranteed NRSTC1MS low-pulse duration is about 1 ms (1 x 32 lsi_ck cycles).
0x2: The guaranteed NRSTC1MS low-pulse duration is about 2 ms (2 x 32 lsi_ck cycles).
...
0x1F:The guaranteed NRSTC1MS low-pulse duration is about 31 ms (31 x 32 lsi_ck cycles).

20.8.33 RCC PWR_LP delay control register (RCC_PWRLPDLYCR)

Address offset: 0x454

Reset value: 0x0000 0000

This register is used to program the delay between the moment where the system exits Stop mode, and the moment where it is allowed to enable the PLLs and to provide a clock to bridges and CPUx. Write access to this register is not allowed during the clock restore sequence (see *Clock restore sequence* for details). Write accesses are defined according to the RCC local resource number 67 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPU2TMPSKP	Res.	Res.	PWRLP_DLY[21:16]					
							rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWRLP_DLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CPU2TMPSKP**: Skip of PWR_LP delay for CPU2

Set and cleared by software. This bit is used by the clock restore sequence to skip the PWR_LP delay for the CPU2 clock restore when the system exits Stop mode.

0: The CPU2 clock restore sequence waits for the PWR_LP delay before activating power consuming elements (default after reset).

1: The CPU2 clock restore sequence runs the PWR_LP delay, but did not wait for the delay to elapse before activating power consuming elements

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:0 **PWRLP_DLY[21:0]**: PWR_LP delay value

Set and cleared by software. This register contains the delay value to be observed before activating the PLL and interconnect clocks, after one of the system Stop modes. The delay is counted with the HSI, or HSI/2 clock.

When LSMCU = 0 in RCC_LSMCUDIVR, the delay value is given by the formula:
PWRLP_DLY[21:0] / FHSI or PWRLP_DLY[21:0] / FHSI / 2

0x0: PWR_LP delay bypassed (default after reset)

0x1: PWR_LP delay of one 64 MHz HSI period (LSMCU = 0) or two HSI periods (LSMCU = 1) is observed.

...

0x3FFFFF: PWR_LP delay of about 65.5 ms (LSMCU = 0) or 131 ms (LSMCU = 1) is observed.

20.8.34 RCC CPU1 clock source interrupt enable set register (RCC_C1CIESETR)

Address offset: 0x458

Reset value: 0x0000 0000

This register is used by the CPU1 to control the interrupt source enable (see *Section 20.5* for details). Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE
											rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLL8RDYIE	PLL7RDYIE	PLL6RDYIE	PLL5RDYIE	PLL4RDYIE	Res.	PLL2RDYIE	PLL1RDYIE	MSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPIE**: CPU1 wake-up from CStop interrupt enable
Set and cleared by software to enable/disable the generation of the interrupt used to wake up the CPU1 from CStop.
0: Wake-up interrupt disabled (default after reset)
1: Wake-up interrupt enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSIE**: LSE clock security system interrupt enable
Set and cleared by software to enable/disable interrupt caused by the CSS on the external 32 kHz oscillator.
0:LSE CSS interrupt disabled (default after reset)
1: LSE CSS interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLL8RDYIE**: PLL8 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL8 lock.
0: PLL8 lock interrupt disabled (default after reset)
1: PLL8 lock interrupt enabled

Bit 11 **PLL7RDYIE**: PLL7 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL7 lock.
0: PLL7 lock interrupt disabled (default after reset)
1: PLL7 lock interrupt enabled

Bit 10 **PLL6RDYIE**: PLL6 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL6 lock.
0: PLL6 lock interrupt disabled (default after reset)
1: PLL6 lock interrupt enabled

Bit 9 **PLL5RDYIE**: PLL5 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL5 lock.
0: PLL5 lock interrupt disabled (default after reset)
1: PLL5 lock interrupt enabled

Bit 8 **PLL4RDYIE**: PLL4 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL4 lock.
0: PLL4 lock interrupt disabled (default after reset)
1: PLL4 lock interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PLL2RDYIE**: PLL2 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled

Bit 5 **PLL1RDYIE**: PLL1 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bit 4 **MSIRDYIE**: MSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.
0: MSI ready interrupt disabled (default after reset)
1: MSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

20.8.35 RCC CPU1 clock source interrupt flag clear register (RCC_C1CIFCLRR)

Address offset: 0x45C

Reset value: 0x0000 0004

This register is used by the CPU1 to read and clear the interrupt flags. Writing 0 has no effect. Writing 1 clears the corresponding flag (see *Section 20.5* for details). Write accesses

are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF
											rc_w1				rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLL8RDYF	PLL7RDYF	PLL6RDYF	PLL5RDYF	PLL4RDYF	Res.	PLL2RDYF	PLL1RDYF	MSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPF**: CPU1 wake-up from CStop interrupt flag
Reset by software by writing this flag to 1. Set by hardware when CPU1 needs to exit CStop.
0: No wake-up interrupt pending (default after reset)
1: Wake-up interrupt pending. Writing 1 clears this flag.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSF**: LSE CSS interrupt flag
Reset by software by writing this flag to 1. Set by hardware when a failure is detected on the external 32 kHz oscillator.
0: No failure detected on the external 32 kHz oscillator (default after reset)
1: A failure is detected on the external 32 kHz oscillator. Writing 1 clears this flag.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLL8RDYF**: PLL8 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL8 locks.
0: No clock ready interrupt caused by PLL8 lock (default after reset)
1: Clock ready interrupt caused by PLL8 lock. Writing 1 clears this flag.

Bit 11 **PLL7RDYF**: PLL7 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL7 locks.
0: No clock ready interrupt caused by PLL7 lock (default after reset)
1: Clock ready interrupt caused by PLL7 lock. Writing 1 clears this flag.

Bit 10 **PLL6RDYF**: PLL6 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL6 locks.
0: No clock ready interrupt caused by PLL6 lock (default after reset)
1: Clock ready interrupt caused by PLL6 lock. Writing 1 clears this flag.

Bit 9 **PLL5RDYF**: PLL5 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL5 locks.
0: No clock ready interrupt caused by PLL5 lock (default after reset)
1: Clock ready interrupt caused by PLL5 lock. Writing 1 clears this flag.

Bit 8 **PLL4RDYF**: PLL4 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL4 locks.
0: No clock ready interrupt caused by PLL4 lock (default after reset)
1: Clock ready interrupt caused by PLL4 lock. Writing 1 clears this flag.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL2 locks.
0: No clock ready interrupt caused by PLL2 lock (default after reset)
1: Clock ready interrupt caused by PLL2 lock. Writing 1 clears this flag.

Bit 5 **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL1 locks.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock. Writing 1 clears this flag.

Bit 4 **MSIRDYF**: MSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when MSI clock becomes stable.
0: No clock ready interrupt caused by the MSI (default after reset)
1: Clock ready interrupt caused by the MSI. Writing 1 clears this flag.

Bit 3 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when HSE clock becomes stable.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE. Writing 1 clears this flag.

Bit 2 **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when HSI clock becomes stable.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI. Writing 1 clears this flag.

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when LSE clock becomes stable.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE. Writing 1 clears this flag.

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when LSI clock becomes stable.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI. Writing 1 clears this flag.

20.8.36 RCC CPU2 clock source interrupt enable set register (RCC_C2CIESETR)

Address offset: 0x460

Reset value: 0x0000 0000

This register is used by the CPU2 to control the interrupt source enable (see *Section 20.5* for more details). Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLL8RDYIE	PLL7RDYIE	PLL6RDYIE	PLL5RDYIE	PLL4RDYIE	Res.	PLL2RDYIE	PLL1RDYIE	MSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPIE**: CPU2 wake-up from CStop Interrupt Enable
Set and cleared by software to enable/disable the generation of the interrupt used to wake up the CPU2 from CStop.
0: Wake-up interrupt disabled (default after reset)
1: Wake-up interrupt enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSIE**: LSE CSS interrupt enable
Set and cleared by software to enable/disable interrupt caused by the CSS on external 32 kHz oscillator.
0: LSE CSS interrupt disabled (default after reset)
1: LSE CSS interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLL8RDYIE**: PLL8 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL8 lock.
0: PLL8 lock interrupt disabled (default after reset)
1: PLL8 lock interrupt enabled

Bit 11 **PLL7RDYIE**: PLL7 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL7 lock.
0: PLL7 lock interrupt disabled (default after reset)
1: PLL7 lock interrupt enabled

Bit 10 **PLL6RDYIE**: PLL6 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL6 lock.
0: PLL6 lock interrupt disabled (default after reset)
1: PLL6 lock interrupt enabled

Bit 9 **PLL5RDYIE**: PLL5 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL5 lock.
0: PLL5 lock interrupt disabled (default after reset)
1: PLL5 lock interrupt enabled

Bit 8 **PLL4RDYIE**: PLL4 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL4 lock.
0: PLL4 lock interrupt disabled (default after reset)
1: PLL4 lock interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PLL2RDYIE**: PLL2 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled

Bit 5 **PLL1RDYIE**: PLL1 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bit 4 **MSIRDYIE**: MSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by MSI stabilization.
0: MSI ready interrupt disabled (default after reset)
1: MSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by HSE stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by HSI stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by LSE stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by LSI stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

20.8.37 RCC CPU2 clock source interrupt flag clear register (RCC_C2CIFCLRR)

Address offset: 0x464

Reset value: 0x0000 0004

This register is used by the CPU2 to read and clear the interrupt flags. Writing 0 has no effect. Writing 1 clears the corresponding flag (see *Section 20.5* for more details).
Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF
											rc_w1				rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLL8RDYF	PLL7RDYF	PLL6RDYF	PLL5RDYF	PLL4RDYF	Res.	PLL2RDYF	PLL1RDYF	MSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPF**: CPU2 wake-up from CStop interrupt flag
Reset by software by writing this flag to 1. Set by hardware when CPU2 needs to exit CStop.
0: No wake-up interrupt pending (default after reset)
1: Wake-up interrupt pending. Writing 1 clears this flag.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSF**: LSE clock security system interrupt flag
Reset by software by writing this flag to 1. Set by hardware when a failure is detected on the external 32 kHz oscillator.
0: No failure detected on the external 32 kHz oscillator (default after reset)
1: A failure is detected on the external 32 kHz oscillator. Writing 1 clears this flag.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLL8RDYF**: PLL8 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL8 locks.
0: No clock ready interrupt caused by PLL8 lock (default after reset)
1: Clock ready interrupt caused by PLL8 lock. Writing 1 clears this flag.

Bit 11 **PLL7RDYF**: PLL7 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL7 locks.
0: No clock ready interrupt caused by PLL7 lock (default after reset)
1: Clock ready interrupt caused by PLL7 lock. Writing 1 clears this flag.

Bit 10 **PLL6RDYF**: PLL6 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL6 locks.
0: No clock ready interrupt caused by PLL6 lock (default after reset)
1: Clock ready interrupt caused by PLL6 lock. Writing 1 clears this flag.

Bit 9 **PLL5RDYF**: PLL5 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL5 locks.
0: No clock ready interrupt caused by PLL5 lock (default after reset)
1: Clock ready interrupt caused by PLL5 lock. Writing 1 clears this flag.

Bit 8 **PLL4RDYF**: PLL4 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL4 locks.
0: No clock ready interrupt caused by PLL4 lock (default after reset)
1: Clock ready interrupt caused by PLL4 lock. Writing 1 clears this flag.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL2 locks.
0: No clock ready interrupt caused by PLL2 lock (default after reset)
1: Clock ready interrupt caused by PLL2 lock. Writing 1 clears this flag.

Bit 5 **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when the PLL1 locks.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock. Writing 1 clears this flag.

Bit 4 **MSIRDYF**: MSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when MSI clock becomes stable.
0: No clock ready interrupt caused by the MSI (default after reset)
1: Clock ready interrupt caused by the MSI. Writing 1 clears this flag.

Bit 3 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when HSE clock becomes stable.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE. Writing 1 clears this flag.

Bit 2 **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when HSI clock becomes stable.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI. Writing 1 clears this flag.

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when LSE clock becomes stable.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE. Writing 1 clears this flag.

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing this flag to 1. Set by hardware when LSI clock becomes stable.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI. Writing 1 clears this flag.

20.8.38 RCC IWDG CPU1 clock freeze set register (RCC_IWDGC1FZSETR)

Address offset: 0x468

Reset value: 0x0000 0000

This register is used by to freeze IWDG1 and IWDG2 clocks. After a system reset, a Standby reset (sys_rstn), or a D1 DStandby reset (vcpu_rstn), the CPU1 can write this register once. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 sets the corresponding bit to 1. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG2**: Freeze of IWDG2 clock
Set once by software after a system, Standby, or D1 DStandby reset, to freeze IWDG2 clock.
0: Writing 0 has no effect. Reading 0 means IWDG2 clock is not frozen (default after reset).
1: Writing 1 freezes IWDG2 clock. Reading 1 means that the IWDG2 clock is frozen.

Bit 0 **FZ_IWDG1**: Freeze of IWDG1 clock
Set once by software after a system, Standby, or D1 DStandby reset, to freeze IWDG1 clock.
0: Writing 0 has no effect. Reading 0 means IWDG1 clock is not frozen (default after reset).
1: Writing 1 freezes IWDG1 clock. Reading 1 means that IWDG1 clock is frozen.

20.8.39 RCC IWDG CPU1 clock freeze clear register (RCC_IWDGC1FZCLRR)

Address offset: 0x46C

Reset value: 0x0000 0000

This register is used to unfreeze IWDG1 and IWDG2 clocks. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG2**: Unfreeze IWDG2 clock
Cleared by software to unfreeze IWDG2 clock.
0: Writing 0 has no effect. Reading 0 means IWDG2 clock is not frozen (default after reset).
1: Writing 1 unfreezes IWDG2 clock. Reading 1 means that IWDG2 clock is frozen.

Bit 0 **FZ_IWDG1**: Unfreeze IWDG1 clock
Cleared by software to unfreeze the IWDG1 clock.
0: Writing 0 has no effect. Reading 0 means IWDG1 clock is not frozen (default after reset).
1: Writing 1 unfreezes IWDG1 clock. Reading 1 means that IWDG1 clock is frozen.

20.8.40 RCC IWDG CPU1 configuration set register (RCC_IWDGC1CFGSETR)

Address offset: 0x470

Reset value: 0x0000 0000

This register is used to enable IWDG1 and IWDG2 timeout event reset behavior, and to activate IWDG2 kernel reset. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 sets the corresponding bit to 1. Wait states are inserted in case of successive write accesses to IWDG1_SYSRSTEN and IWDG2_SYSRSTEN. The number of wait states may be up to 7 cycles of the RCC clock. IWDG1_SYSRSTEN

and IWDG2_SYSRSTEN are reset by app_rstn reset, and are located in V_{DD} domain. IWDG2_KERRST is reset by sys_rstn reset. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_KERRST	Res.	Res.
													rs		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_SYSRSTEN	Res.	IWDG1_SYSRSTEN
													rs		rs

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **IWDG2_KERRST**: IWDG2 kernel reset

Set by software.

0: Writing 0 has no effect. Reading 0 means that the IWDG2 kernel reset is not active.

1: Writing 1 activates IWDG2 kernel reset Reading 1 means IWDG2 kernel reset is active.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **IWDG2_SYSRSTEN**: Enable of IWDG2 timeout event to reset the application

Set by software.

0: Writing 0 has no effect. Reading 0 means that IWDG2 timeout event cannot trigger an application reset.

1: Writing 1 enables IWDG2 timeout event to reset the application. Reading 1 means that IWDG2 timeout event triggers an application reset.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IWDG1_SYSRSTEN**: Enable of IWDG1 timeout event to reset the application

Set by software.

0: Writing 0 has no effect. Reading 0 means that IWDG1 timeout event does not trigger an application reset when CPU1 is not the trusted agent.

1: Writing 1 enables IWDG1 timeout event to reset the application when CPU1 is not the trusted agent. Reading 1 means that IWDG1 timeout event triggers an application reset when CPU1 is not the trusted agent.

20.8.41 RCC IWDG CPU1 configuration clear register (RCC_IWDGC1CFGCLRR)

Address offset: 0x474

Reset value: 0x0000 0000

This register is used to disable IWDG1 and IWDG2 timeout event reset behavior, and to deactivate IWDG2 kernel reset. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. Wait states are inserted in case of successive write accesses to IWDG1_SYSRSTEN and IWDG2_SYSRSTEN. The number of wait states may be up to 7 cycles of RCC clock. Write accesses are defined according to the RCC local resource number 70 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_KERRST	Res.	Res.
													rc_w1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_SYSRSTEN	Res.	IWDG1_SYSRSTEN
													rc_w1		rc_w1

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **IWDG2_KERRST**: IWDG2 kernel reset

Set by software.

0: Writing 0 has no effect. Reading 0 means that IWDG2 kernel reset is not active

1: Writing 1 deactivates IWDG2 kernel reset. Reading 1 means IWDG2 kernel reset is active.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **IWDG2_SYSRSTEN**: Clear of IWDG2 timeout event to reset the application

Set by software.

0: Writing 0 has no effect. Reading 0 means that IWDG2 timeout event cannot trigger an application reset.

1: Writing 1 clears IWDG2 timeout event to reset the application. Reading 1 means that IWDG2 timeout event is enabled.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IWDG1_SYSRSTEN**: Clear of IWDG1 timeout event to reset the application

Set by software.

0: Writing 0 has no effect. Reading 0 means that IWDG1 timeout event cannot trigger an application reset when CPU1 is not the trusted agent.

1: Writing 1 clears IWDG1 timeout event to reset the application when CPU1 is not the trusted agent. Reading 1 means that IWDG1 timeout event triggers an application reset when CPU1 is not the trusted agent.

20.8.42 RCC IWDG CPU2 clock freeze set register (RCC_IWDGC2FZSETR)

Address offset: 0x478

Reset value: 0x0000 0000

This register is used by to freeze IWDG3 and IWDG4 clocks. After a system reset, or a Standby reset (sys_rstn), the CPU2 can write this register once. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 sets the corresponding bit to 1. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG4	FZ_IWDG3
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG4**: Freeze of IWDG4 clock

Set once by software after a system or Standby reset to freeze IWDG4 clock.
0: Writing 0 has no effect. Reading 0 means IWDG4 clock is not frozen (default after reset).
1: Writing 1 freezes IWDG4 clock. Reading 1 means that IWDG4 clock is frozen.

Bit 0 **FZ_IWDG3**: Freeze of IWDG3 clock

Set once by software after a system or Standby reset to freeze IWDG3 clock.
0: Writing 0 has no effect. Reading 0 means IWDG3 clock is not frozen (default after reset).
1: Writing 1 freezes IWDG3 clock. Reading 1 means that IWDG3 clock is frozen.

20.8.43 RCC IWDG CPU2 clock freeze clear register (RCC_IWDGC2FZCLRR)

Address offset: 0x47C

Reset value: 0x0000 0000

This register is used to unfreeze IWDG3 and IWDG4 clocks. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG4	FZ_IWDG3
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG4**: Unfreeze of IWDG4 clock
Cleared by software to unfreeze IWDG4 clock.
0: Writing 0 has no effect. Reading 0 means that the IWDG4 clock is not frozen (default after reset).
1: Writing 1 unfreezes the IWDG4 clock. Reading 1 means that the IWDG4 clock is frozen.

Bit 0 **FZ_IWDG3**: Unfreeze of IWDG3 clock
Cleared by software to unfreeze IWDG3 clock.
0: Writing 0 has no effect. Reading 0 means IWDG3 clock is not frozen (default after reset).
1: Writing 1 unfreezes IWDG3 clock. Reading 1 means that IWDG3 clock is frozen.

20.8.44 RCC IWDG CPU2 configuration set register (RCC_IWDGC2CFGSETR)

Address offset: 0x480

Reset value: 0x0000 0000

This register is used to enable IWDG3 and IWDG4 timeout event reset behavior, and to activate IWDG4 kernel reset. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 sets the corresponding bit to 1. Wait states are inserted in case of successive write accesses to IWDG3_SYSRSTEN and IWDG4_SYSRSTEN. The number of wait states may be up to 7 cycles of the RCC clock. IWDG3_SYSRSTEN and IWDG4_SYSRSTEN are reset by app_rstn reset, and are located in the V_{DD} domain. IWDG4_KERRST is reset by sys_rstn reset. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_KERRST	Res.	Res.
													rs		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_SYSRSTEN	Res.	IWDG3_SYSRSTEN
													rs		rs

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **IWDG4_KERRST**: IWDG4 kernel reset
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG4 kernel reset is not active.
1: Writing 1 activates IWDG4 kernel reset. Reading 1 means IWDG4 kernel reset is active.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **IWDG4_SYSRSTEN**: Enable of IWDG4 timeout event to reset the application
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG4 timeout event cannot trigger an application reset.
1: Writing 1 enables IWDG4 timeout event to reset the application. Reading 1 means that IWDG4 timeout event triggers an application reset.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IWDG3_SYSRSTEN**: Enable of IWDG3 timeout event to reset the application
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG3 timeout event does not trigger an application reset when CPU1 is not the trusted agent.
1: Writing 1 enables IWDG3 timeout event to reset the application when CPU1 is not the trusted agent. Reading 1 means that IWDG3 timeout event triggers an application reset when CPU1 is not the trusted agent.

20.8.45 RCC IWDG CPU2 configuration clear register (RCC_IWDGC2CFGCLRR)

Address offset: 0x484

Reset value: 0x0000 0000

This register is used to disable IWDG3 and IWDG4 timeout event reset behavior, and to deactivate IWDG4 kernel reset. Writing 0 has no effect. Reading returns the effective values of the corresponding bits. Writing 1 clears the corresponding bit to 0. Wait states are inserted in case of successive write accesses to IWDG3_SYSRSTEN and IWDG4_SYSRSTEN. The number of wait states may be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 71 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_KERRST	Res.	Res.
													rc_w1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_SYSRSTEN	Res.	IWDG3_SYSRSTEN
													rc_w1		rc_w1

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **IWDG4_KERRST**: IWDG4 kernel reset
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG4 kernel reset is not active.
1: Writing 1 deactivates IWDG4 kernel reset. Reading 1 means IWDG4 kernel reset is active.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **IWDG4_SYSRSTEN**: Clear of IWDG4 timeout event to reset the application
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG4 timeout event cannot trigger an application reset.
1: Writing 1 clears IWDG4 timeout event to reset the application. Reading 1 means that IWDG4 timeout event is enabled.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IWDG3_SYSRSTEN**: Clear of IWDG3 timeout event to reset the application
Set by software.
0: Writing 0 has no effect. Reading 0 means that IWDG3 timeout event cannot trigger an application reset when CPU1 is not the trusted agent.
1: Writing 1 clears IWDG3 timeout event to reset the application when CPU1 is not the trusted agent. Reading 1 means that IWDG3 timeout event triggers an application reset when CPU1 is not the trusted agent.

20.8.46 RCC MCO1 configuration register (RCC_MCO1CFGR)

Address offset: 0x488

Reset value: 0x0000 0000

This register is used to select the clock generated on the MCO1 output. Write accesses are defined according to the RCC local resource number 108 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO1 ON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO1 SEL
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **MCO1ON**: Control of MCO1 output
Set and cleared by software.
0: MCO1 output disabled (default after reset)
1: MCO1 output enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **MCO1SEL**: MCO1 clock output selection
Set and cleared by the software. Clock source selection may generate glitches on MCO1. It is highly recommended to configure this bit only after reset, before enabling the external oscillators and the PLLs.
0: Channel 61 output of the flexclkgen engine (default after reset)
1: Clock observation 0 of the flexclkgen engine

20.8.47 RCC MCO2 configuration register (RCC_MCO2CFGR)

Address offset: 0x48C

Reset value: 0x0000 0000

This register is used to select the clock generated on the MCO2 output. Write accesses are defined according to the RCC local resource number 109 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO2 ON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO2 SEL
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **MCO2ON**: Control of the MCO2 output
Set and cleared by software.
0: MCO2 output disabled (default after reset)
1: MCO2 output enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **MCO2SEL**: MCO2 clock output selection
Set and cleared by software. Clock source selection may generate glitches on MCO2. It is highly recommended to configure this bit only after reset, before enabling the external oscillators and the PLLs.
0: Channel 62 output of the flexclkgen engine (default after reset)
1: Clock observation 1 of the flexclkgen engine

20.8.48 RCC oscillator clock enable set register (RCC_OCENSETR)

Address offset: 0x490

Reset value: 0x0000 0001

This register is used to control the oscillators. Writing 0 has no effect. Writing 1 sets the corresponding bits. Reading gives the effective values of each bit. Write access to this register is not allowed during the clock restore sequence (see *Clock restore sequence* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSEKERON	HSEON	HSEDIGBYP	HSEDIV2BYP	HSEDIV2ON	Res.	MSIKERON	MSION	HSIKERON	HSION
				rs	rs	rs	rs	rs	rs	rs		rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **HSECSSON**: HSECSSON bit set
Set CSS on HSE. This bit is set-only (disabled by system reset, or when the system enters Standby). When this bit is set, the clock detector is enabled by hardware when HSE is ready.
0: Reading 0 means that CSS on HSE is off (default after reset).
1: Writing 1 enables CSS on HSE. Reading 1 means that CSS on HSE is on.

Bit 10 **HSEBYP**: HSEBYP bit set
Set by software in order to enable the HSE bypass mode.
0: No effect
1: Set the HSEBYP bit.

Bit 9 **HSEKERON**: HSEKERON bit set
Set by software to force HSE on even in Stop mode, to be quickly available as kernel clock for some peripherals. This bit has no effect on HSEON value.
0: No effect
1: Set the HSEKERON bit.

Bit 8 **HSEON**: HSEON bit set
Set by software to enable HSE.
0: No effect
1: Set HSEON bit.

Bit 7 **HSEDIGBYP**: HSEDIGBYP bit set
Set by software when the external clock connected to OSC_IN is a full-swing digital signal.
0: No effect
1: Set HSEDIGBYP bit (digital bypass).

Bit 6 **HSEDIV2BYP**: HSEDIV2BYP bit set
Set by software to bypass the HSE divide-by-2 clock.
0: No effect
1: Set HSEDIV2BYP bit.

Bit 5 **HSEDIV2ON**: HSEDIV2ON bit set
Set by software to enable the HSE divide-by-2 clock.
0: No effect
1: Set HSEDIV2ON bit.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **MSIKERON**: MSIKERON bit set
Set by software to force MSI on even in Stop mode, to be quickly available as kernel clock for peripherals. This bit has no effect on MSION value.
0: No effect
1: Set the MSIKERON bit.

Bit 2 **MSION**: MSION bit set
Set by software to enable MSI.
0: No effect
1: Set the MSION bit.

Bit 1 **HSIKERON**: HSIKERON bit set
Set by software to force HSI on even in Stop mode, to be quickly available as kernel clock for peripherals. This bit has no effect on HSION value.
0: No effect
1: Set the HSIKERON bit.

Bit 0 **HSION**: HSION bit set

Set by software to enable HSI. This bit is also set by hardware to force HSI on when the product exits Standby or Stop, or in case of a failure of HSE which is used as the system clock source.

0: No effect

1: Set the HSION bit.

20.8.49 RCC oscillator clock enable clear register (RCC_OCENCLRR)

Address offset: 0x494

Reset value: 0x0000 0001

This register is used to control the oscillators. Writing 0 has no effect. Writing 1 clears the corresponding bits. Reading gives the effective values of the enable bits. Write access to this register is not allowed during the clock restore sequence (see *Clock restore sequence* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSEBYP	HSEKERON	HSEON	HSEDIGBYP	HSEDIV2BYP	HSEDIV2ON	Res.	MSIKERON	MSION	HSIKERON	HSION
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSEBYP**: HSEBYP bit clear

Cleared by software in order to disable the bypass mode.

0: No effect

1: Clear the HSEBYP bit.

Bit 9 **HSEKERON**: HSEKERON bit clear

Cleared by software to allow the HSE to be switched off in Stop mode.

0: No effect

1: Clear the HSEKERON bit.

Bit 8 **HSEON**: HSEON bit clear

Cleared by software to disable the HSE clock.

0: No effect

1: Clear the HSEON bit.

Bit 7 **HSEDIGBYP**: HSEDIGBYP bit clear

Cleared by software when the external clock connected to OSC_IN is a low-swing signal.

0: No effect

1: Clear the HSEDIGBYP bit (analog bypass).

Bit 6 **HSEDIV2BYP**: HSEDIV2BYP bit clear

Cleared by software to use the HSE divide-by-2 clock.

0: No effect

1: Clear the HSEDIV2BYP bit.

Bit 5 **HSEDIV2ON**: HSEDIV2ON bit clear
Cleared by software to disable the HSE divide-by-2 clock.
0: No effect
1: Clear HSEDIV2ON bit.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **MSIKERON**: MSIKERON bit clear
Cleared by software to allow the MSI to be switched off in Stop mode. This bit has no effect on MSION value.
0: No effect
1: Clear the MSIKERON bit.

Bit 2 **MSION**: MSION bit clear
Cleared by software to disable the MSI clock.
0: No effect
1: Clear the MSION bit.

Bit 1 **HSIKERON**: HSIKERON bit clear
Cleared by software to allow the HSI to be switched off in Stop mode. This bit has no effect on HSION value.
0: No effect
1: Clear the HSIKERON bit.

Bit 0 **HSION**: HSION bit clear
Cleared by software to disable the HSI clock.
0: No effect
1: Clear the HSION bit.

20.8.50 RCC oscillator clock ready register (RCC_OCRDYR)

Address offset: 0x498

Reset value: 0x0000 0001

This is a read-only access register that contains the status flags of the oscillators. Writing has no effect. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CKRESTT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	Res.	Res.	Res.	Res.	Res.	MSIRDY	Res.	HSIRDY
							r						r		r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CKREST**: Clock restore state machine status
Set by hardware to indicate if the clock restore sequence is on-going.
0: Clock restore process not ongoing (default after reset)
1: Clock restore process ongoing

Bits 24:9 Reserved, must be kept at reset value.

Bit 8 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock not ready (default after reset)
1: HSE clock ready

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **MSIRDY**: MSI clock ready flag
Set by hardware to indicate that the MSI oscillator is stable.
0: MSI clock not ready (default after reset)
1: MSI clock ready

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HSIRDY**: HSI clock ready flag
Set by hardware to indicate that the HSI oscillator is stable.
0: HSI clock not ready (default after reset)
1: HSI clock ready

20.8.51 RCC HSI configuration register (RCC_HSICFGR)

Address offset: 0x49C

Reset value: 0x0000 0000

This register is used to fine tune the HSI frequency. Write access is not allowed during the clock restore sequence (see *Clock restore sequence* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSICAL[8:0]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **HSICAL[8:0]**: HSI clock calibration
Set by hardware when option-byte loading during system reset. Adjusted by software through HSITRIM trimming bits.
This bitfield contains the sum of engineering option-byte calibration and HSITRIM values.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust calibration. This bitfield represents a signed value. It is added to engineering option bytes loaded during reset phase (bsec_hsi_cal[8:0]) to form calibration trimming value: HSICAL[8:0] = HSITRIM[6:0] + bsec_hsi_cal[8:0].

0x40: bsec_hsi_cal[8:0] - 64
0x41: bsec_hsi_cal[8:0] - 63
...
0x0: bsec_hsi_cal[8:0] (default after reset)
...
0x3E: bsec_hsi_cal[8:0] + 62
0x3F: bsec_hsi_cal[8:0] + 63

Bits 7:0 Reserved, must be kept at reset value.

20.8.52 RCC MSI configuration register (RCC_MSICFGR)

Address offset: 0x4A0

Reset value: 0x0010 1000

This register is used to fine tune the MSI frequency. Write access is not allowed during the clock restore sequence (see *Clock restore sequence* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSICAL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	MSITRIM[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **MSICAL[7:0]**: MSI clock calibration

Set by hardware when option-byte loading during system reset. Adjusted by software through MSITRIM trimming bits.

This bit-field contains the sum of engineering option-byte calibration and MSITRIM values.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **MSITRIM[4:0]**: MSI clock trimming

Set by software to adjust calibration. This bitfield is added to engineering option bytes loaded during reset phase (bsec_msi_cal[7:0]) to form the calibration trimming value.

MSICAL[7:0] = MSITRIM[4:0] + bsec_msi_cal[7:0].

Bits 7:0 Reserved, must be kept at reset value.

20.8.53 RCC LSI control register (RCC_LSICR)

Address offset: 0x4A4

Reset value: 0x0000 0000

This register is used to control the LSI. This register is reset when a system reset occurs (app_rstn), but not when the circuit exits from Standby. Wait states are inserted in case of successive write accesses to this register. The number of wait states can be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
														r	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready
Set and cleared by hardware to indicate when the LSI is stable. This bit needs 3 cycles of lsi_ck to fall down after LSION has been cleared to 0.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 0 **LSION**: LSI oscillator enabled
Set and cleared by software.
0: LSI oscillator off
1: LSI oscillator on

20.8.54 RCC RTC clock divider register (RCC_RTCDIVR)

Address offset: 0x4A8

Reset value: 0x0000 0000

This register is used to divide the HSE clock for the RTC. Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDIV[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RTCDIV[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE clock for RTC.

Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 4 MHz. These bits must be configured if needed before selecting the RTC clock source.

Note: When RTCSRC not equal to 3, this divider is disabled (does not provide clock).

0x0: HSE (default after reset)
0x1: HSE/2
0x2: HSE/3
0x3: HSE/4
...
0x3E:HSE/63
0x3F:HSE/64

20.8.55 RCC APB1 clock divider register (RCC_APB1DIVR)

Address offset: 0x4AC

Reset value: 0x8000 0000

This register is used to control the APB1 clock prescaler (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB1DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB1DIV[2:0]		
													rw	rw	rw

Bit 31 **APB1DIVRDY**: APB1 clock prescaler status

Set and cleared by hardware to indicate if the new division factor is taken into account.
0: The new division factor is not yet taken into account.
1: The new division factor is taken into account (default after reset).

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB1DIV[2:0]**: APB1 clock divider

Set and cleared by software to control the APB1 clock division factor. Changing this division ratio impacts the APB1 clock domain frequency. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APB1DIV update. The application can check if the new division factor is taken into account by reading APB1DIVRDY.

0x0: ck_icn_ls_mcu (default after reset)
0x1: ck_icn_ls_mcu / 2
0x2: ck_icn_ls_mcu / 4
0x3: ck_icn_ls_mcu / 8
0x4: ck_icn_ls_mcu / 16
Others: Reserved

20.8.56 RCC APB2 clock divider register (RCC_APB2DIVR)

Address offset: 0x4B0

Reset value: 0x8000 0000

This register is used to control the APB2 clock prescaler (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB2DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB2DIV[2:0]		
													rw	rw	rw

Bit 31 **APB2DIVRDY**: APB2 clock prescaler status

Set and cleared by hardware to indicate if the new division factor is taken into account.
0: The new division factor is not yet taken into account.
1: The new division factor is taken into account (default after reset).

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB2DIV[2:0]**: APB2 clock divider

Set and cleared by software to control APB2 clock division factor. Changing this division ratio impacts the APB2 clock domain frequency. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APB2DIV update. The application can check if the new division factor is taken into account by reading APB2DIVRDY.
0x0: ck_icn_ls_mcu (default after reset)
0x1: ck_icn_ls_mcu / 2
0x2: ck_icn_ls_mcu / 4
0x3: ck_icn_ls_mcu / 8
0x4: ck_icn_ls_mcu / 16
Others: Reserved

20.8.57 RCC APB3 clock divider register (RCC_APB3DIVR)

Address offset: 0x4B4

Reset value: 0x8000 0000

This register is used to control the APB3 clock prescaler (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB3DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIV[2:0]		
													rw	rw	rw

Bit 31 **APB3DIVRDY**: APB3 clock prescaler status

Set and cleared by hardware to indicate if the new division factor is taken into account.
0: The new division factor is not yet taken into account.
1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB3DIV[2:0]**: APB3 clock divider

Set and cleared by software to control APB3 clock division factor.
Changing this division ratio has an impact on the frequency of the APB3 clock domain. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APB3DIV update. The application can check if the new division factor is taken into account by reading APB3DIVRDY.
0x0: ck_icn_ls_mcu (default after reset)
0x1: ck_icn_ls_mcu / 2
0x2: ck_icn_ls_mcu / 4
0x3: ck_icn_ls_mcu / 8
0x4: ck_icn_ls_mcu / 16
Others: Reserved

20.8.58 RCC APB4 clock divider register (RCC_APB4DIVR)

Address offset: 0x4B8

Reset value: 0x8000 0000

This register is used to control the APB4 clock divider (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB4DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB4DIV[2:0]		
													rw	rw	rw

Bit 31 **APB4DIVRDY**: APB4 clock divider status

Set and cleared by hardware to indicate if the new division factor is taken into account.
0: The new division factor is not yet taken into account.
1: The new division factor is taken into account (default after reset).

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB4DIV[2:0]**: APB4 clock divider

Set and cleared by software to control APB4 clock division factor. This division ratio can be changed on-the-fly. It impacts only the APB4 clock frequency. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APB4DIV update.
The application can check if the new division factor is taken into account by reading APB4DIVRDY.
0x0: ck_icn_ls_mcu (default after reset)
0x1: ck_icn_ls_mcu / 2
0x2: ck_icn_ls_mcu / 4
0x3: ck_icn_ls_mcu / 8
0x4: ck_icn_ls_mcu / 16
Others: Reserved

20.8.59 RCC APB5 clock divider register (RCC_APB5DIVR)

Address offset: 0x4BC

Reset value: 0x8000 0000

This register is used to control the APB5 clock divider (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB5DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB5DIV[2:0]		
													rw	rw	rw

Bit 31 **APB5DIVRDY**: APB5 clock divider status

Set and cleared by hardware to indicate if the new division factor is taken into account.
0: The new division factor is not yet taken into account.
1: The new division factor is taken into account (default after reset).

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB5DIV[2:0]**: APB5 clock divider

Set and cleared by software to control APB5 clock division factor. This division ratio can be changed on-the-fly. It impacts only the APB5 clock frequency. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APB5DIV update.
The application can check if the new division factor is taken into account by reading APB5DIVRDY.
0x0: ck_icn_ls_mcu (default after reset)
0x1: ck_icn_ls_mcu / 2
0x2: ck_icn_ls_mcu / 4
0x3: ck_icn_ls_mcu / 8
0x4: ck_icn_ls_mcu / 16
Others: Reserved

20.8.60 RCC APBDBG clock divider register (RCC_APBDBGDIVR)

Address offset: 0x4C0

Reset value: 0x8000 0000

This register is used to control the APBDBG clock divider (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APBDBGDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBDBGDIV[2:0]		
													rw	rw	rw

Bit 31 **APBDBGDIVRDY**: APBDBG clock divider status

Set and cleared by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account (default after reset).

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APBDBGDIV[2:0]**: APBDBG clock divider

Set and cleared by software to control APBDBG clock division factor. This division ratio can be changed on-the-fly. It impacts only the APBDBG clock frequency. Clocks are divided with the new prescaler factor, from 1 to 16 ck_icn_ls_mcu cycles after APBDBGDIV update. The application can check if the new division factor is taken into account by reading APBDBGDIVRDY.

0x0: ck_icn_ls_mcu (default after reset)

0x1: ck_icn_ls_mcu / 2

0x2: ck_icn_ls_mcu / 4

0x3: ck_icn_ls_mcu / 8

0x4: ck_icn_ls_mcu / 16

Others: Reserved

20.8.61 RCC TIM group 1 prescaler register (RCC_TIMG1PRER)

Address offset: 0x4C8

Reset value: 0x8000 0000

This register is used to control the prescaler value of timers located in the APB1 domain: TIM2 to TIM7, and TIM10 to TIM14 (see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMG1PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG1PRE
															rw

Bit 31 **TIMG1PRERDY**: Clock prescaler status for timers of group 1

Set and cleared by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account (default after reset).

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **TIMG1PRE**: Clock prescaler selection for timers of group 1

This bit is set and cleared by software to control the clock frequency of all timers connected to APB1 domain (see *Table 126)*.

0: Timer kernel clock is equal to ck_icn_ls_mcu if APB1DIV corresponds to a division by 1 or 2. Otherwise it is equal to 2 x Fck_icn_apb1 (default after reset).

1: Timer kernel clock is equal to ck_icn_ls_mcu if APB1DIV corresponds to a division by 1, 2, or 4. Otherwise, it is equal to 4 x Fck_icn_apb1.

20.8.62 RCC TIM group 2 prescaler register (RCC_TIMG2PRER)

Address offset: 0x4CC

Reset value: 0x8000 0000

This register is used to control the prescaler value of timers located in the APB2 domain: TIM1, TIM8, TIM15, TIM16, TIM17 (see *Root clock generation* for details).
Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMG2PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG2PRE
															rw

Bit 31 **TIMG2PRERDY**: Clock prescaler status for timers of group 2

Set and cleared by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **TIMG2PRE**: Clock prescaler selection for timers of group 2

This bit is set and cleared by software to control the clock frequency of all timers connected to APB2 domain (see *Table 126)*.

0: Timer kernel clock is equal to ck_icn_ls_mcu if APB2DIV corresponds to a division by 1 or 2. Otherwise it is equal to 2 x Fck_icn_apb2 (default after reset).

1: Timer kernel clock is equal to ck_icn_ls_mcu if APB2DIV corresponds to division by 1, 2, or 4.Otherwise it is equal to 4 x Fck_icn_apb2.

20.8.63 RCC LSMCU clock divider register (RCC_LSMCUDIVR)

Address offset: 0x4D0

Reset value: 0x8000 0000

This register is used to control the LSMU (low-speed MCU clock divider, see *Root clock generation* for details). Write accesses are defined according to the RCC local resource number 69 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSMCUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSMCUDIV
															rw

Bit 31 **LSMCUDIVRDY**: LSMCU clock divider status

Set and cleared by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account (default after reset).

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **LSMCUDIV**: LSMCU clock divider

Set and cleared by software to control ck_icn_ls_mcu clock division factor. This division ratio can be changed on-the-fly. It impacts only the ck_icn_ls_mcu clock frequency. This clock is divided with the new prescaler factor, from 1 to 2 ck_icn_hs_mcu cycles after LSMCUDIV update. The application can check if the new division factor is taken into account by reading LSMCUDIVRDY.

0x0: ck_icn_hs_mcu (default after reset)

0x1: ck_icn_hs_mcu / 2

20.8.64 RCC DDRCTRL AXI port configuration register (RCC_DDRCPCFGR)

Address offset: 0x4D4

Reset value: 0x0000 0005

This register is used to control reset, clock gating, and low-power behavior of the DDRCTRL AXI ports. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRC PLPEN	DDRC PEN	DDRC PRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DDRCPLPEN**: DDRCTRL AXI port clock enable during CSleep

Set and cleared by software. This bit controls ck_icn_s_ddr clock gating in CSleep.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DDRCPEN**: DDRCTRL AXI port clock enable

Set and cleared by software. This bit control sck_icn_s_ddr clock gating.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DDRCPRST**: DDRCTRL AXI port reset

Set and cleared by software.

0: Writing 0 releases DDRCTRL AXI port interface reset. Reading 0 means that DDRCTRL AXI port interface is not under reset (default after reset).

1: Writing 1 sets DDRCTRL AXI port interface under reset. Reading 1 means that DDRCTRL AXI port interface is under reset.

20.8.65 RCC DDRCTRL APB configuration register (RCC_DDRCAPBCFGR)

Address offset: 0x4D8

Reset value: 0x0000 0005

This register is used to control reset, clock gating, and low-power behavior of the DDRCTRL APB. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRC APBLP EN	DDRC APBEN	DDRC APBRS T
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DDRCAPBLPEN**: DDRCTRL APB clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enbled in CSleep.

Bit 1 **DDRCAPBEN**: DDRCTRL APB clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DDRCAPBRST**: DDRCTRL APB reset

Set and cleared by software.

0: Writing 0 releases DDRCTRL APB interface reset. Reading 0 means that DDRCTRL APB interface is not under reset (default after reset).

1: Writing 1 sets DDRCTRL APB interface under reset. Reading 1 means that DDRCTRL APB interface is under reset.

20.8.66 RCC DDRPHYC APB configuration register (RCC_DDRPHYCAPBCFGR)

Address offset: 0x4DC

Reset value: 0x0000 0005

This register is used to control reset, clock gating, and low-power behavior of the DDRPHYC APB interface. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPHYCAPBLPEN	DDRPHYCAPBEN	DDRPHYCAPBRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DDRPHYCAPBLPEN**: DDRPHYC APB clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DDRPHYCAPBEN**: DDRPHYC APB clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DDRPHYCAPBRST**: DDRPHYC APB reset

Set and cleared by software.

0: Writing 0 releases DDRPHYC APB interface reset. Reading 0 means that DDRPHYC APB interface is not under reset (default after reset).

1: Writing 1 sets DDRPHYC APB interface under reset. Reading 1 means that DDRPHYC APB interface is under reset.

20.8.67 RCC DDRPHYC configuration register (RCC_DDRPHYCCFGR)

Address offset: 0x4E0

Reset value: 0x0000 0000

This register is used to control clock gating behavior of the DDRPHYC. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRP HYCEN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DDRPHYCEN**: DDRPHYC kernel clock enable
Set and cleared by software.
0: Writing 0 disables DDRPHYC kernel clock. Reading 0 means that DDRPHYC kernel clock is disabled.
1: Writing 1 enables DDRPHYC kernel clock. Reading 1 means that DDRPHYC kernel clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.68 RCC DDR configuration register (RCC_DDRCFGR)

Address offset: 0x4E4

Reset value: 0x0000 0005

This register is used to control reset, clock gating, and low-power behavior of the DDR APB configuration. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRCF GLPEN	DDRCF GEN	DDRCF GRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DDRCFGLPEN**: DDR APB configuration clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DDRCFGEN**: DDR APB configuration clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DDRCFGRST**: DDR APB configuration reset
Set and cleared by software.
0: Writing 0 releases DDR APB configuration interface reset. Reading 0 means that DDR APB configuration interface is not under reset (default after reset).
1: Writing 1 sets DDR APB configuration interface under reset. Reading 1 means that DDR APB configuration interface is under reset.

20.8.69 RCC DDR interface configuration register (RCC_DDRITFCFGR)

Address offset: 0x4E8

Reset value: 0x0000 0001

This register is used to control the DDR interface according to supported low-power modes. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPHYDLP
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRSHR	Res.	Res.	DDRCKMOD[1:0]		Res.	Res.	Res.	DDRRST
							rw			rw	rw				rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DDRPHYDLP**: DDRPHY deep low-power mode (LP2 or LP3)
Set and cleared by software.
0: DDRPHY not set in low-power mode (default after reset)
1: DDRPHY set in deep low-power mode

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRSHR**: DDR peripheral mode
Set and cleared by software.
0: DDR defined as single peripheral (default after reset)
1: DDR defined as shared peripheral

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DDRCKMOD[1:0]**: RCC mode for DDR clock control

Set and cleared by software.

0x0: Normal mode: dphy_ker_ck and dphy_ker_byp_ck gating depends on DDRPHYCEN, DDRSHR, DDRPHYDLP, and CPU1 or system mode. ddrc_ker_ck gating depends on DDRCPEN, DDRCPLPEN, DDRSHR, and CPU1 or system mode. This mode must be selected during DDRCTRL and DDRPHYC initialization phase, and if the application uses SSR (software self-refresh) mode.

0x1: ASR (automatic self-refresh) mode: ddrc_ker_ck is gated automatically according to cactive_ddrc signal. dphy_ker_ck and dphy_ker_byp_ck must be kept running in this mode.

0x2: HSR (hardware self-refresh) mode: ddrc_ker_ck gating depends on DDRCPEN, DDRCLPEN, DDRSHR, and CPU1 or system mode. The dphy_ker_ck and dphy_ker_byp_ck must be kept running in this mode.

Others: Reserved

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DDRRST**: DDR core and PHY reset

Set and cleared by software.

0: DDR core and PHY are not reset.

1: DDR core and PHY are reset (default after reset).

20.8.70 RCC SYSRAM configuration register (RCC_SYSRAMCFGR)

Address offset: 0x4F0

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the SYSRAM. Write accesses are defined according to the RCC local resource number 74 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN	SYSRAMEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SYSRAMLPEN**: SYSRAM clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SYSRAMEN**: SYSRAM clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.71 RCC SRAM1 configuration register (RCC_SRAM1CFGR)

Address offset: 0x4F8

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the SRAM1. Write accesses are defined according to the RCC local resource number 78 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1 LPEN	SRAM1 EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SRAM1LPEN**: SRAM1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SRAM1EN**: SRAM1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.72 RCC RETRAM configuration register (RCC_RETRAMCFGR)

Address offset: 0x500

Reset value: 0x0000 0006

This register is used to control clock gating and low-power behavior of the RETRAM. Write accesses are defined according to the RCC local resource number 76 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRA MLPEN	RETRA MEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **RETRAMLPEN**: RETRAM clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **RETRAMEN**: RETRAM clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.73 RCC BKPSRAM configuration register (RCC_BKPSRAMCFGR)

Address offset: 0x504

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the BKPSRAM. Write accesses are defined according to the RCC local resource number 77 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	BKPSRAMEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **BKPSRAMLPEN**: BKPSRAM clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **BKPSRAMEN**: BKPSRAM clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.74 RCC OCTOSPI1 configuration register (RCC_OSPI1CFGR)

Address offset: 0x514

Reset value: 0x0000 0004

This register is used to control the reset, clock gating, and low-power behavior of the OCTOSPI1 and the OTFDEC1. Write accesses are defined according to the RCC local resource number 110 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI1 DLLRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC1RST	Res.	Res.	Res.	Res.	Res.	OSPI1LPEN	OSPI1EN	OSPI1RST
							rw						rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **OSPI1DLLRST**: OCTOSPI1 DLL reset

Set and cleared by software.

0: Writing 0 releases OCTOSPI1 DLL reset. Reading 0 means that OCTOSPI1 DLL reset is released.

1: Writing 1 asserts OCTOSPI1 DLL reset. Reading 1 means that OCTOSPI1 DLL reset is asserted.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OTFDEC1RST**: OTFDEC1 reset

Set and cleared by software.

0: Writing 0 releases OTFDEC1 reset. Reading 0 means that OTFDEC1 reset is released.

1: Writing 1 asserts OTFDEC1 reset. Reading 1 means that OTFDEC1 reset is asserted.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **OSPI1LPEN**: OCTOSPI1, OTFDEC1 and OCTOSPI1 delay (DLYBOSPI1) clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **OSPI1EN**: OCTOSPI1 and OTFDEC1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **OSPI1RST**: OCTOSPI1 reset

Set and cleared by software.

0: Writing 0 releases OCTOSPI1 reset. Reading 0 means that OCTOSPI1 reset is released.

1: Writing 1 asserts OCTOSPI1 reset. Reading 1 means that OCTOSPI1 reset is asserted.

20.8.75 RCC FMC configuration register (RCC_FMCCFGR)

Address offset: 0x51C

Reset value: 0x0000 0004

This register is used to control the reset, clock gating, and low-power behavior of the FMC. Write accesses are defined according to the RCC local resource number 112 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCLPEN	FMCEN	FMCRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FMCLPEN**: FMC clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **FMCEN**: FMC clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **FMCRST**: FMC reset

Set and cleared by software.

0: Writing 0 releases FMC reset. Reading 0 means that FMC reset is released.

1: Writing 1 asserts FMC reset. Reading 1 means that FMC reset is asserted.

20.8.76 RCC debug configuration register (RCC_DBGCFGR)

Address offset: 0x520

Reset value: 0x0000 0000

This is register contains the enable control of debug and trace functions. Write accesses are defined according to the RCC local resource number 73 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBGRST	Res.	DBGMCUEN	TRACEEN	DBGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw		rw	rw	rw								

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **DBGRST**: Debug function reset
Set and cleared by software to control the reset generation for trace and all debug.
0: Trace and debug parts are not reset (default after reset).
1: Trace and debug parts are under reset.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **DBGMCUEN**: Clock enable for dbgmcu
Set and cleared by software to control ck_icn_p_dbgmcu and ck_dap generation.
0: Clock for dbgmcu function disabled (default after reset)
1: Clock for dbgmcu function enabled

Bit 9 **TRACEEN**: Clock enable for trace port interface
Set and cleared by software to control ck_ker_tpiu generation.
0: Clock for trace function disabled (default after reset)
1: Clock for trace function enabled

Bit 8 **DBGEN**: Debug function clock enable
Set and cleared by software to control ck_icn_apbdbg and ck_sys_atb generation.
0: Enabling of clocks for debug function is controlled by cdbgpwrupreq signal from DAP (default after reset).
1: Clocks for debug function are enabled.

Bits 7:0 Reserved, must be kept at reset value.

20.8.77 RCC STM configuration register (RCC_STMCFGR)

Address offset: 0x524

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the STM. Write accesses are defined according to the RCC local resource number 73 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STMLPEN	STMEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **STMLPEN**: STM clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **STMEN**: STM clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.78 RCC ETR configuration register (RCC_ETRCFGR)

Address offset: 0x528

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the Cortex-A35 ETR interface. Write accesses are defined according to the RCC local resource number 73 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRLPEN	ETREN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ETRLPEN**: ETR clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **ETREN**: ETR clocks enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.79 RCC GPIOx configuration register (RCC_GPIOxCFGR) (x = A to I)

Address offset: Block A: 0x52C

Address offset: Block B: 0x530

Address offset: Block C: 0x534

Address offset: Block D: 0x538

Address offset: Block E: 0x53C

Address offset: Block F: 0x540

Address offset: Block G: 0x544

Address offset: Block H: 0x548

Address offset: Block I: 0x54C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the GPIOx. Write accesses are defined according to the RCC local resource number 90 to 98 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOx LPEN	GPIOx EN	GPIOx RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **GPIOxLPEN**: GPIO x clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **GPIOxEN**: GPIO x clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **GPIOxRST**: GPIO x reset

Set and cleared by software.

0: Writing 0 releases GPIOx reset. Reading 0 means that GPIOx reset is released.

1: Writing 1 asserts GPIOx reset. Reading 1 means that GPIOx reset is asserted.

20.8.80 RCC GPIOZ configuration register (RCC_GPIOZCFGR)

Address offset: 0x558

Reset value: 0x0000 0004

This register is used to control reset, clock gating, low-power, and autonomous clock mode behavior of the GPIOZ. Wait states are inserted in case of successive write accesses to this register, except on GPIOZRST. The number of wait states may be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 101 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOZ LPEN	GPIOZ EN	GPIOZ RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **GPIOZLPEN**: GPIOZ clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **GPIOZEN**: GPIOZ clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **GPIOZRST**: GPIOZ reset

Set and cleared by software.

0: Writing 0 releases GPIOZ reset. Reading 0 means that GPIOZ reset is released.

1: Writing 1 asserts GPIOZ reset. Reading 1 means that GPIOZ reset is asserted.

20.8.81 RCC HPDMA1 configuration register (RCC_HPDMA1CFGR)

Address offset: 0x55C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the HPDMA1. Write accesses are defined according to the RCC local resource number 83 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA1LPEN	HPDMA1EN	HPDMA1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HPDMA1LPEN**: HPDMA1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **HPDMA1EN**: HPDMA1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HPDMA1RST**: HPDMA1 reset

Set and cleared by software.

0: Writing 0 releases HPDMA1 reset. Reading 0 means that HPDMA1 reset is released.

1: Writing 1 asserts HPDMA1 reset. Reading 1 means that HPDMA1 reset is asserted.

20.8.82 RCC HPDMA2 configuration register (RCC_HPDMA2CFGR)

Address offset: 0x560

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the HPDMA2. Write accesses are defined according to the RCC local resource number 84 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA2LPEN	HPDMA2EN	HPDMA2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HPDMA2LPEN**: HPDMA2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **HPDMA2EN**: HPDMA2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HPDMA2RST**: HPDMA2 reset

Set and cleared by software.

0: Writing 0 releases HPDMA2 reset. Reading 0 means that HPDMA2 reset is released.

1: Writing 1 asserts HPDMA2 reset. Reading 1 means that HPDMA2 reset is asserted.

20.8.83 RCC HPDMA3 configuration register (RCC_HPDMA3CFGR)

Address offset: 0x564

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the HPDMA3. Write accesses are defined according to the RCC local resource number 85 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA3LPEN	HPDMA3EN	HPDMA3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HPDMA3LPEN**: HPDMA3 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **HPDMA3EN**: HPDMA3 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HPDMA3RST**: HPDMA3 reset

Set and cleared by software.

0: Writing 0 releases HPDMA3 reset. Reading 0 means that HPDMA3 reset is released.

1: Writing 1 asserts HPDMA3 reset. Reading 1 means that HPDMA3 reset is asserted.

20.8.84 RCC IPCC1 configuration register (RCC_IPCC1CFGR)

Address offset: 0x570

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of IPCC1. Write accesses are defined according to the RCC local resource number 87 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCC1LPEN	IPCC1EN	IPCC1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IPCC1LPEN**: IPCC1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **IPCC1EN**: IPCC1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **IPCC1RST**: IPCC1 reset

Set and cleared by software.

0: Writing 0 releases IPCC1 reset. Reading 0 means that IPCC1 reset is released.

1: Writing 1 asserts IPCC1 reset. Reading 1 means that IPCC1 reset is asserted.

20.8.85 RCC RTC configuration register (RCC_RTCCFGR)

Address offset: 0x578

Reset value: 0x0000 0004

This register is used to control clock gating, low-power, and autonomous clock mode behavior of the RTC. Wait states are inserted in case of successive write accesses to this register. The number of wait states may be up to 7 cycles of the RCC clock. Write accesses are defined according to the RCC local resource number 102 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCLPEN	RTCEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **RTCLPEN**: RTC bus clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.
1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clocks is enabled in CSleep.

Bit 1 **RTCEN**: RTC bus clock enable
Set and cleared by software.
0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.
1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.86 RCC SYSCPU1 configuration register (RCC_SYSCPU1CFGR)

Address offset: 0x580

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the CPU1 system configuration. Write accesses are defined according to the RCC local resource number 106 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCPU1LPEN	SYSCPU1EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SYSCPU1LPEN**: CPU1 system configuration clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.
1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clocks is enabled in CSleep.

Bit 1 **SYSCPU1EN**: CPU1 system configuration clock enable
Set and cleared by software.
0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.
1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.87 RCC BSEC configuration register (RCC_BSECCFGR)

Address offset: 0x584

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of the BSEC. Write accesses are defined according to the RCC local resource number 103 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSECLPEN	BSECEN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **BSECLPEN**: BSEC clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.
1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clocks is enabled in CSleep.

Bit 1 **BSECEN**: BSEC clock enable
Set and cleared by software.
0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.
1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.88 RCC PLL2 configuration register 1 (RCC_PLL2CFGR1)

Address offset: 0x590

Reset value: 0x0000 0001

This register is used to configure PLL2 in the device. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CKREFST	Res.	Res.	Res.	PLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rc_w				r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSMODRST
							rw								rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CKREFST**: PLL2 reference clock state

Set by hardware and cleared by software. Indicates if PLL2 reference clock is running. After any reset, this bit is updated after 5 clock cycles of the selected PLL reference clock.

0: Reading 0 means that PLL2 reference clock is not running. Writing 0 resets the reference clock running detector.

1: Reading 1 means that PLL2 reference clock is running. Writing 1 resets the reference clock running detector.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **PLLRDY**: PLL2 clock ready flag

Set by hardware to indicate when the PLL2 is locked

0: PLL2 unlocked (default after reset)

1: PLL2 locked

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **PLLEN**: PLL2 enable

Set and cleared by software to enable PLL2.

0: PLL2 off (default after reset)

1: PLL2 on

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SSMODRST**: PLL2 spread-spectrum modulator reset

Set and cleared by software.

0: PLL2 spread-spectrum modulator reset module is released.

1: PLL2 spread-spectrum modulator reset module is asserted (default after reset).

20.8.89 RCC PLL2 configuration register 2 (RCC_PLL2CFGR2)

Address offset: 0x594

Reset value: 0x0000 0000

This register is used to configure PLL2 in the device. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FBDIV[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **FBDIV[11:0]**: PLL2 VCO multiplication factor

Set and cleared by software to control the VCO multiplication factor.
In integer mode, this value must be set between 16 and 640.
In fractional mode, this value must be set between 20 and 320.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **FREFDIV[5:0]**: PLL2 reference input clock divide frequency ratio

Set and cleared by software.
0x0: Not applicable when PLL2 is enabled
0x1: FREF divided by 1 (min value)
0x2: FREF divided by 2
0x3: FREF divided by 3
...
0x3C: FREF divided by 60
0x3D: FREF divided by 61
0x3E: FREF divided by 62
0x3F: FREF divided by 63

20.8.90 RCC PLL2 configuration register 3 (RCC_PLL2CFGR3)

Address offset: 0x598

Reset value: 0x0400 0000

This register is used to configure the PLL2 in the device. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SSCGDIS	DACEN	DOWNSPREAD	FRACIN[23:16]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SSCGDIS**: PLL2 spread-spectrum modulator enable

Set and cleared by software to enable the PLL2 clock spreading generator.
0: PLL2 spread-spectrum modulator enabled
1: PLL2 spread-spectrum modulator disabled (default after reset)

Bit 25 **DACEN**: PLL2 noise canceling DAC enable in fractional mode

Set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 24 **DOWNSPREAD**: PLL2 VCO frequency modulation mode

Set and cleared by software to select the PLL2 clock spreading mode.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bits 23:0 **FRACIN[23:0]**: Fractional part of PLL2 VCO multiplication factor

Set and cleared by software to control the VCO fractional multiplication factor.
VCO output frequency = FREF * (FBDIV + FRAC / 2^{24}) / FREFDIV

20.8.91 RCC PLL2 configuration register 4 (RCC_PLL2CFGR4)

Address offset: 0x59C

Reset value: 0x0000 0000

This register is used to configure PLL2 in the device. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BYPASS	FOUTPOSTDIVEN	DSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw								

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **BYPASS**: PLL2 FOUTPOSTDIV bypass
Set and cleared by software to bypass FOUTPOSTDIV by FREF.
0: FOUTPOSTDIV driven by POSTDIV2
1: FOUTPOSTDIV driven by FREF

Bit 9 **FOUTPOSTDIVEN**: PLL2 output and post divider enable
Set and cleared by software to enable PLL2 output and post dividers.
0: FOUTPOSTDIV output, POSTDIV1, and POSTDIV2 are powered down.
1: FOUTPOSTDIV output, POSTDIV1, and POSTDIV2 dividers are active.

Bit 8 **DSMEN**: PLL2 delta-sigma modulator enable
Set and cleared by software to enable the delta-sigma modulator (DSM).
0: DSM is powered down.
1: DSM is active.

Bits 7:0 Reserved, must be kept at reset value.

20.8.92 RCC PLL2 configuration register 5 (RCC_PLL2CFGR5)

Address offset: 0x5A0

Reset value: 0x0000 0000

This register is used to configure PLL2 in the device. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPREAD[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVVAL[3:0]			
												rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **SPREAD[4:0]**: Modulation depth adjustment for PLL2
Set and cleared by software to adjust the modulation depth of clock spreading generator.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **DIVVAL[3:0]**: Modulation frequency adjustment for PLL2
Set and cleared by software to adjust the modulation frequency of clock spreading generator.

20.8.93 RCC PLL2 configuration register 6 (RCC_PLL2CFGR6)

Address offset: 0x5A8

Reset value: 0x0000 0000

This register is used to configure PLL2 in the product. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV1[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **POSTDIV1[2:0]**: PLL2 VCO frequency divide level 1
Set and cleared by software to divide the VCO frequency output.
0x0: Not applicable
0x1: FOUTVCO divided by 1 (minimum value)
...
0x6: FOUTVCO divided by 6
0x7: FOUTVCO divided by 7

20.8.94 RCC PLL2 configuration register 7 (RCC_PLL2CFGR7)

Address offset: 0x5AC

Reset value: 0x0000 0000

This register is used to configure PLL2 in the product. Write accesses are defined according to the RCC local resource number 104 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **POSTDIV2[2:0]**: PLL2 VCO frequency divide level 2

Set and cleared by software to divide the VCO frequency output.

0x0: Not applicable

0x1: FOUTVCO/POSTDIV1 divided by 1 (minimum value)

...

0x6: FOUTVCO/POSTDIV1 divided by 6

0x7: FOUTVCO/POSTDIV1 divided by 7

*Note: FOUTPOSTDIV = FOUTVCO / POSTDIV1 * POSTDIV2*

20.8.95 RCC HSI frequency monitoring control register (RCC_HSIFMONCR)

Address offset: 0x5E0

Reset value: 0x001F 07A1

This register is used to control the HSI frequency monitoring module. Write accesses are defined according to the RCC local resource number 113 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSIMONF	HSIMONIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIDEV[5:0]					
rc_w1	rw									rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSIMONEN	Res.	Res.	Res.	Res.	HSIREF[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HSIMONF**: HSI clock period monitor interrupt flag

Reset by software by writing 1 to this flag. Set by hardware when a deviation is detected on HSI clock period.

0: No deviation detected on HSI clock period (default after reset)

1: Deviation detected on HSI clock period. Writing 1 clears this flag.

Bit 30 **HSIMONIE**: HSI clock period monitor interrupt enable

Set and cleared by software to enable/disable interrupt caused by HSI clock period monitoring.

0: HSI clock period monitoring interrupt disabled (default after reset)

1: HSI clock period monitoring interrupt enabled

Bits 29:22 Reserved, must be kept at reset value.

Bits 21:16 **HSIDEV[5:0]**: HSI clock count deviation value

Set and cleared by software.

Bit 15 **HSIMONEN**: HSI clock period monitor enable

Set and cleared by software.

0: Writing 0 disables HSI clock period monitoring. Reading 0 means that HSI clock period monitoring is disabled.

1: Writing 1 enables HSI clock period monitoring. Reading 1 means that HSI clock period monitoring is enabled.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **HSIREF[10:0]**: HSI clock count reference value
Set and cleared by software.

20.8.96 RCC HSI frequency value register (RCC_HSIFVALR)

Address offset: 0x5E4

Reset value: 0x0000 0000

This read-only access register contains the number of HSI clock cycles measured between consecutive edges of LSE. Writing has no effect. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSIVAL[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **HSIVAL[10:0]**: HSI clock-cycle counter value
Set by hardware.

20.8.97 RCC MSI frequency monitoring control register (RCC_MSIFMONCR)

Address offset: 0x5E8

Reset value: 0x000F 01E8

This register is used to control the MSI frequency monitoring module. Write accesses are defined according to the RCC local resource number 88 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIMONF	MSIMONIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSIDEV[4:0]				
rc_w1	rw										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIMONEN	Res.	Res.	Res.	Res.	Res.	Res.	MSIREF[8:0]								
rw							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSIMONF**: MSI clock period monitor interrupt flag
Reset by software by writing 1 to this flag. Set by hardware when a deviation is detected on MSI clock period.
0: No deviation detected on MSI clock period (default after reset)
1: Deviation detected on MSI clock period. Writing 1 clears this flag.

Bit 30 **MSIMONIE**: MSI clock period monitor interrupt enable
Set and cleared by software to enable/disable interrupt caused by MSI clock period monitoring.
0: MSI clock period monitoring interrupt disabled (default after reset)
1: MSI clock period monitoring interrupt enabled

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:16 **MSIDEV[4:0]**: MSI clock count deviation value
Set and cleared by software.

Bit 15 **MSIMONEN**: MSI clock period monitor enable
Set and cleared by software.
0: Writing 0 disables MSI clock period monitoring. Reading 0 means that MSI clock period monitoring is disabled.
1: Writing 1 enables MSI clock period monitoring. Reading 1 means that MSI clock period monitoring is enabled.

Bits 14:9 Reserved, must be kept at reset value.

Bits 8:0 **MSIREF[8:0]**: MSI clock count reference value
Set and cleared by software.

20.8.98 RCC MSI frequency value register (RCC_MSIFVALR)

Address offset: 0x5EC

Reset value: 0x0000 0000

This read-only access register contains the number of MSI clock cycles measured between consecutive edges of LSE. Writing has no effect. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSIVAL[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **MSIVAL[8:0]**: MSI clock-cycle counter value
Set by hardware.

20.8.99 RCC TIM1 configuration register (RCC_TIM1CFGR)

Address offset: 0x700

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM1LPEN	TIM1EN	TIM1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM1LPEN**: TIM1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM1EN**: TIM1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM1RST**: TIM1 reset

Set and cleared by software.

0: Writing 0 releases TIM1 reset. Reading 0 means that TIM1 reset is released.

1: Writing 1 asserts TIM1 reset. Reading 1 means that TIM1 reset is asserted.

20.8.100 RCC TIM2 configuration register (RCC_TIM2CFGR)

Address offset: 0x704

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2LPEN	TIM2EN	TIM2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM2LPEN**: TIM2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM2EN**: TIM2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM2RST**: TIM2 reset
Set and cleared by software.
0: Writing 0 releases TIM2 reset. Reading 0 means that TIM2 reset is released.
1: Writing 1 asserts TIM2 reset. Reading 1 means that TIM2 reset is asserted.

20.8.101 RCC TIM3 configuration register (RCC_TIM3CFGR)

Address offset: 0x708

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3LPEN	TIM3EN	TIM3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM3LPEN**: TIM3 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM3EN**: TIM3 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM3RST**: TIM3 reset
Set and cleared by software.
0: Writing 0 releases TIM3 reset. Reading 0 means that TIM3 reset is released.
1: Writing 1 asserts TIM3 reset. Reading 1 means that TIM3 reset is asserted.

20.8.102 RCC TIM4 configuration register (RCC_TIM4CFGR)

Address offset: 0x70C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM4LPEN	TIM4EN	TIM4RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM4LPEN**: TIM4 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM4EN**: TIM4 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM4RST**: TIM4 reset

Set and cleared by software.

0: Writing 0 releases TIM4 reset. Reading 0 means that TIM4 reset is released.

1: Writing 1 asserts TIM4 reset. Reading 1 means that TIM4 reset is asserted.

20.8.103 RCC TIM5 configuration register (RCC_TIM5CFGR)

Address offset: 0x710

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM5. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM5LPEN	TIM5EN	TIM5RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM5LPEN**: TIM5 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM5EN**: TIM5 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM5RST**: TIM5 reset
Set and cleared by software.
0: Writing 0 releases TIM5 reset. Reading 0 means that TIM5 reset is released.
1: Writing 1 asserts TIM5 reset. Reading 1 means that TIM5 reset is asserted.

20.8.104 RCC TIM6 configuration register (RCC_TIM6CFGR)

Address offset: 0x714

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM6. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM6LPEN	TIM6EN	TIM6RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM6LPEN**: TIM6 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM6EN**: TIM6 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM6RST**: TIM6 reset
Set and cleared by software.
0: Writing 0 releases TIM6 reset. Reading 0 means that TIM6 reset is released.
1: Writing 1 asserts TIM6 reset. Reading 1 means that TIM6 reset is asserted.

20.8.105 RCC TIM7 configuration register (RCC_TIM7CFGR)

Address offset: 0x718

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM7. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7LPEN	TIM7EN	TIM7RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM7LPEN**: TIM7 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM7EN**: TIM7 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM7RST**: TIM7 reset

Set and cleared by software.

0: Writing 0 releases TIM7 reset. Reading 0 means that TIM7 reset is released.

1: Writing 1 asserts TIM7 reset. Reading 1 means that TIM7 reset is asserted.

20.8.106 RCC TIM8 configuration register (RCC_TIM8CFGR)

Address offset: 0x71C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM8. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM8LPEN	TIM8EN	TIM8RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM8LPEN**: TIM8 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM8EN**: TIM8 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM8RST**: TIM8 reset

Set and cleared by software.

0: Writing 0 releases TIM8 reset. Reading 0 means that TIM8 reset is released.

1: Writing 1 asserts TIM8 reset. Reading 1 means that TIM8 reset is asserted.

20.8.107 RCC TIM10 configuration register (RCC_TIM10CFGR)

Address offset: 0x720

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM10. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM10LPEN	TIM10EN	TIM10RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM10LPEN**: TIM10 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM10EN**: TIM10 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM10RST**: TIM10 reset

Set and cleared by software.

0: Writing 0 releases TIM10 reset. Reading 0 means that TIM10 reset is released.

1: Writing 1 asserts TIM10 reset. Reading 1 means that TIM10 reset is asserted.

20.8.108 RCC TIM11 configuration register (RCC_TIM11CFGR)

Address offset: 0x724

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM11. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM11LPEN	TIM11EN	TIM11RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM11LPEN**: TIM11 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM11EN**: TIM11 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM11RST**: TIM11 reset

Set and cleared by software.

0: Writing 0 releases TIM11 reset. Reading 0 means that TIM11 reset is released.

1: Writing 1 asserts TIM11 reset. Reading 1 means that TIM11 reset is asserted.

20.8.109 RCC TIM12 configuration register (RCC_TIM12CFGR)

Address offset: 0x728

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM12. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM12LPEN	TIM12EN	TIM12RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM12LPEN**: TIM12 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM12EN**: TIM12 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM12RST**: TIM12 reset

Set and cleared by software.

0: Writing 0 releases TIM12 reset. Reading 0 means that TIM12 reset is released.

1: Writing 1 asserts TIM12 reset. Reading 1 means that TIM12 reset is asserted.

20.8.110 RCC TIM13 configuration register (RCC_TIM13CFGR)

Address offset: 0x72C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM13. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM13LPEN	TIM13EN	TIM13RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM13LPEN**: TIM13 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM13EN**: TIM13 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM13RST**: TIM13 reset

Set and cleared by software.

0: Writing 0 releases TIM13 reset. Reading 0 means that TIM13 reset is released.

1: Writing 1 asserts TIM13 reset. Reading 1 means that TIM13 reset is asserted.

20.8.111 RCC TIM14 configuration register (RCC_TIM14CFGR)

Address offset: 0x730

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM14. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM14LPEN	TIM14EN	TIM14RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM14LPEN**: TIM14 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM14EN**: TIM14 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM14RST**: TIM14 reset

Set and cleared by software.

0: Writing 0 releases TIM14 reset. Reading 0 means that TIM14 reset is released.

1: Writing 1 asserts TIM14 reset. Reading 1 means that TIM14 reset is asserted.

20.8.112 RCC TIM15 configuration register (RCC_TIM15CFGR)

Address offset: 0x734

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM15. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM15LPEN	TIM15EN	TIM15RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM15LPEN**: TIM15 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM15EN**: TIM15 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM15RST**: TIM15 reset
Set and cleared by software.
0: Writing 0 releases TIM15 reset. Reading 0 means that TIM15 reset is released.
1: Writing 1 asserts TIM15 reset. Reading 1 means that TIM15 reset is asserted.

20.8.113 RCC TIM16 configuration register (RCC_TIM16CFGR)

Address offset: 0x738

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM16. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM16LPEN	TIM16EN	TIM16RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM16LPEN**: TIM16 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM16EN**: TIM16 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM16RST**: TIM16 reset
Set and cleared by software.
0: Writing 0 releases TIM16 reset. Reading 0 means that TIM16 reset is released.
1: Writing 1 asserts TIM16 reset. Reading 1 means that TIM16 reset is asserted.

20.8.114 RCC TIM17 configuration register (RCC_TIM17CFGR)

Address offset: 0x73C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the TIM17. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17LPEN	TIM17EN	TIM17RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TIM17LPEN**: TIM17 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **TIM17EN**: TIM17 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **TIM17RST**: TIM17 reset

Set and cleared by software.

0: Writing 0 releases TIM17 reset. Reading 0 means that TIM17 reset is released.

1: Writing 1 asserts TIM17 reset. Reading 1 means that TIM17 reset is asserted.

20.8.115 RCC LPTIM1 configuration register (RCC_LPTIM1CFGR)

Address offset: 0x744

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the LPTIM1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1LPEN	LPTIM1EN	LPTIM1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPTIM1LPEN**: LPTIM1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPTIM1EN**: LPTIM1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPTIM1RST**: LPTIM1 reset
Set and cleared by software.
0: Writing 0 releases LPTIM1 reset. Reading 0 means that LPTIM1 reset is released.
1: Writing 1 asserts LPTIM1 reset. Reading 1 means that LPTIM1 reset is asserted.

20.8.116 RCC LPTIM2 configuration register (RCC_LPTIM2CFGR)

Address offset: 0x748

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the LPTIM2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2 LPEN	LPTIM2 EN	LPTIM2 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPTIM2LPEN**: LPTIM2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPTIM2EN**: LPTIM2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPTIM2RST**: LPTIM2 reset
Set and cleared by software.
0: Writing 0 releases LPTIM2 reset. Reading 0 means that LPTIM2 reset is released.
1: Writing 1 asserts LPTIM2 reset. Reading 1 means that LPTIM2 reset is asserted.

20.8.117 RCC LPTIM3 configuration register (RCC_LPTIM3CFGR)

Address offset: 0x74C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, low-power behavior of the LPTIM3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM3 LPEN	LPTIM3 EN	LPTIM3 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPTIM3LPEN**: LPTIM3 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPTIM3EN**: LPTIM3 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPTIM3RST**: LPTIM3 reset

Set and cleared by software.

0: Writing 0 releases LPTIM3 reset. Reading 0 means that LPTIM3 reset is released.

1: Writing 1 asserts LPTIM3 reset. Reading 1 means that LPTIM3 reset is asserted.

20.8.118 RCC LPTIM4 configuration register (RCC_LPTIM4CFGR)

Address offset: 0x750

Reset value: 0x0000 0004

This register is used to control reset, clock gating, low-power behavior of the LPTIM4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM4 LPEN	LPTIM4 EN	LPTIM4 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPTIM4LPEN**: LPTIM4 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPTIM4EN**: LPTIM4 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPTIM4RST**: LPTIM4 reset
Set and cleared by software.
0: Writing 0 releases LPTIM4 reset. Reading 0 means that LPTIM4 reset is released.
1: Writing 1 asserts LPTIM4 reset. Reading 1 means that LPTIM4 reset is asserted.

20.8.119 RCC LPTIM5 configuration register (RCC_LPTIM5CFGR)

Address offset: 0x754

Reset value: 0x0000 0004

This register is used to control reset, clock gating, low-power behavior of the LPTIM5. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM5 LPEN	LPTIM5 EN	LPTIM5 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPTIM5LPEN**: LPTIM5 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPTIM5EN**: LPTIM5 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPTIM5RST**: LPTIM5 reset
Set and cleared by software.
0: Writing 0 releases LPTIM5 reset. Reading 0 means that LPTIM5 reset is released.
1: Writing 1 asserts LPTIM5 reset. Reading 1 means that LPTIM5 reset is asserted.

20.8.120 RCC SPI1 configuration register (RCC_SPI1CFGR)

Address offset: 0x758

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI1LPEN	SPI1EN	SPI1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI1LPEN**: SPI1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI1EN**: SPI1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI1RST**: SPI1 reset

Set and cleared by software.

0: Writing 0 releases SPI1 reset. Reading 0 means that SPI1 reset is released.

1: Writing 1 asserts SPI1 reset. Reading 1 means that SPI1 reset is asserted.

20.8.121 RCC SPI2 configuration register (RCC_SPI2CFGR)

Address offset: 0x75C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2LPEN	SPI2EN	SPI2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI2LPEN**: SPI2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI2EN**: SPI2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI2RST**: SPI2 reset

Set and cleared by software.

0: Writing 0 releases SPI2 reset. Reading 0 means that SPI2 reset is released.

1: Writing 1 asserts SPI2 reset. Reading 1 means that SPI2 reset is asserted.

20.8.122 RCC SPI3 configuration register (RCC_SPI3CFGR)

Address offset: 0x760

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3LPEN	SPI3EN	SPI3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI3LPEN**: SPI3 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI3EN**: SPI3 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI3RST**: SPI3 reset

Set and cleared by software.

0: Writing 0 releases SPI3 reset. Reading 0 means that SPI3 reset is released.

1: Writing 1 asserts SPI3 reset. Reading 1 means that SPI3 reset is asserted.

20.8.123 RCC SPI4 configuration register (RCC_SPI4CFGR)

Address offset: 0x764

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI4LPEN	SPI4EN	SPI4RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI4LPEN**: SPI4 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI4EN**: SPI4 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI4RST**: SPI4 reset

Set and cleared by software.

0: Writing 0 releases SPI4 reset. Reading 0 means that SPI4 reset is released.

1: Writing 1 asserts SPI4 reset. Reading 1 means that SPI4 reset is asserted.

20.8.124 RCC SPI5 configuration register (RCC_SPI5CFGR)

Address offset: 0x768

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI5. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI5LPEN	SPI5EN	SPI5RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI5LPEN**: SPI5 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI5EN**: SPI5 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI5RST**: SPI5 reset
Set and cleared by software.
0: Writing 0 releases SPI5 reset. Reading 0 means that SPI5 reset is released.
1: Writing 1 asserts SPI5 reset. Reading 1 means that SPI5 reset is asserted.

20.8.125 RCC SPI6 configuration register (RCC_SPI6CFGR)

Address offset: 0x76C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPI6. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI6LPEN	SPI6EN	SPI6RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPI6LPEN**: SPI6 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPI6EN**: SPI6 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPI6RST**: SPI6 reset
Set and cleared by software.
0: Writing 0 releases SPI6 reset. Reading 0 means that SPI6 reset is released.
1: Writing 1 asserts SPI6 reset. Reading 1 means that SPI6 reset is asserted.

20.8.126 RCC SPDIFRX configuration register (RCC_SPDIFRXCFGR)

Address offset: 0x778

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SPDIFRX. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPDIFRXLPEN	SPDIFRXEN	SPDIFRXRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SPDIFRXLPEN**: SPDIFRX clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SPDIFRXEN**: SPDIFRX clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SPDIFRXRST**: SPDIFRX reset

Set and cleared by software.

0: Writing 0 releases SPDIFRX reset. Reading 0 means that SPDIFRX reset is released.

1: Writing 1 asserts SPDIFRX reset. Reading 1 means that SPDIFRX reset is asserted.

20.8.127 RCC USART1 configuration register (RCC_USART1CFGR)

Address offset: 0x77C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USART1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1LPEN	USART1EN	USART1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USART1LPEN**: USART1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USART1EN**: USART1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USART1RST**: USART1 reset
Set and cleared by software.
0: Writing 0 releases USART1 reset. Reading 0 means that USART1 reset is released.
1: Writing 1 asserts USART1 reset. Reading 1 means that USART1 reset is asserted.

20.8.128 RCC USART2 configuration register (RCC_USART2CFGR)

Address offset: 0x780

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USART2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART 2LPEN	USART 2EN	USART 2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USART2LPEN**: USART2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USART2EN**: USART2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USART2RST**: USART2 reset
Set and cleared by software.
0: Writing 0 releases USART2 reset. Reading 0 means that USART2 reset is released.
1: Writing 1 asserts USART2 reset. Reading 1 means that USART2 reset is asserted.

20.8.129 RCC USART3 configuration register (RCC_USART3CFGR)

Address offset: 0x784

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USART3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART 3LPEN	USART 3EN	USART 3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USART3LPEN**: USART3 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USART3EN**: USART3 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USART3RST**: USART3 reset

Set and cleared by software.

0: Writing 0 releases USART3 reset. Reading 0 means that USART3 reset is released.

1: Writing 1 asserts USART3 reset. Reading 1 means that USART3 reset is asserted.

20.8.130 RCC UART4 configuration register (RCC_UART4CFGR)

Address offset: 0x788

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the UART4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART4 LPEN	UART4 EN	UART4 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **UART4LPEN**: UART4 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **UART4EN**: UART4 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **UART4RST**: UART4 reset
Set and cleared by software.
0: Writing 0 releases UART4 reset. Reading 0 means that UART4 reset is released.
1: Writing 1 asserts UART4 reset. Reading 1 means that UART4 reset is asserted.

20.8.131 RCC UART5 configuration register (RCC_UART5CFGR)

Address offset: 0x78C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the UART5. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART5 LPEN	UART5 EN	UART5 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **UART5LPEN**: UART5 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **UART5EN**: UART5 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **UART5RST**: UART5 reset
Set and cleared by software.
0: Writing 0 releases UART5 reset. Reading 0 means that UART5 reset is released.
1: Writing 1 asserts UART5 reset. Reading 1 means that UART5 reset is asserted.

20.8.132 RCC USART6 configuration register (RCC_USART6CFGR)

Address offset: 0x790

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USART6. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART 6LPEN	USART 6EN	USART 6RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USART6LPEN**: USART6 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USART6EN**: USART6 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USART6RST**: USART6 reset

Set and cleared by software.

0: Writing 0 releases USART6 reset. Reading 0 means that USART6 reset is released.

1: Writing 1 asserts USART6 reset. Reading 1 means that USART6 reset is asserted.

20.8.133 RCC UART7 configuration register (RCC_UART7CFGR)

Address offset: 0x794

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the UART7. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART7 LPEN	UART7 EN	UART7 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **UART7LPEN**: UART7 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **UART7EN**: UART7 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **UART7RST**: UART7 reset
Set and cleared by software.
0: Writing 0 releases UART7 reset. Reading 0 means that UART7 reset is released.
1: Writing 1 asserts UART7 reset. Reading 1 means that UART7 reset is asserted.

20.8.134 RCC LPUART1 configuration register (RCC_LPUART1CFGR)

Address offset: 0x7A0

Reset value: 0x0000 0004

This register is used to control reset, clock gating, low-power behavior of the LPUART1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPUART1LPEN	LPUART1EN	LPUART1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LPUART1LPEN**: LPUART1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LPUART1EN**: LPUART1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LPUART1RST**: LPUART1 reset
Set and cleared by software.
0: Writing 0 releases LPUART1 reset. Reading 0 means that LPUART1 reset is released.
1: Writing 1 asserts LPUART1 reset. Reading 1 means that LPUART1 reset is asserted.

20.8.135 RCC I2C1 configuration register (RCC_I2C1CFGR)

Address offset: 0x7A4

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the I2C1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1LPEN	I2C1EN	I2C1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I2C1LPEN**: I2C1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I2C1EN**: I2C1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I2C1RST**: I2C1 reset

Set and cleared by software.

0: Writing 0 releases I2C1 reset. Reading 0 means that I2C1 reset is released.

1: Writing 1 asserts I2C1 reset. Reading 1 means that I2C1 reset is asserted.

20.8.136 RCC I2C2 configuration register (RCC_I2C2CFGR)

Address offset: 0x7A8

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the I2C2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2LPEN	I2C2EN	I2C2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I2C2LPEN**: I2C2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I2C2EN**: I2C2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I2C2RST**: I2C2 reset
Set and cleared by software.
0: Writing 0 releases I2C2 reset. Reading 0 means that I2C2 reset is released.
1: Writing 1 asserts I2C2 reset. Reading 1 means that I2C2 reset is asserted.

20.8.137 RCC I2C3 configuration register (RCC_I2C3CFGR)

Address offset: 0x7AC

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the I2C3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3LPEN	I2C3EN	I2C3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I2C3LPEN**: I2C3 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I2C3EN**: I2C3 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I2C3RST**: I2C3 reset
Set and cleared by software.
0: Writing 0 releases I2C3 reset. Reading 0 means that I2C3 reset is released.
1: Writing 1 asserts I2C3 reset. Reading 1 means that I2C3 reset is asserted.

20.8.138 RCC SAI1 configuration register (RCC_SAI1CFGR)

Address offset: 0x7C4

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SAI1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1LPEN	SAI1EN	SAI1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SAI1LPEN**: SAI1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SAI1EN**: SAI1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SAI1RST**: SAI1 reset

Set and cleared by software.

0: Writing 0 releases SAI1 reset. Reading 0 means that SAI1 reset is released.

1: Writing 1 asserts SAI1 reset. Reading 1 means that SAI1 reset is asserted.

20.8.139 RCC SAI2 configuration register (RCC_SAI2CFGR)

Address offset: 0x7C8

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SAI2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPEN	SAI2EN	SAI2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SAI2LPEN**: SAI2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SAI2EN**: SAI2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SAI2RST**: SAI2 reset
Set and cleared by software.
0: Writing 0 releases SAI2 reset. Reading 0 means that SAI2 reset is released.
1: Writing 1 asserts SAI2 reset. Reading 1 means that SAI2 reset is asserted.

20.8.140 RCC SAI3 configuration register (RCC_SAI3CFGR)

Address offset: 0x7CC

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SAI3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI3LPEN	SAI3EN	SAI3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SAI3LPEN**: SAI3 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SAI3EN**: SAI3 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SAI3RST**: SAI3 reset
Set and cleared by software.
0: Writing 0 releases SAI3 reset. Reading 0 means that SAI3 reset is released.
1: Writing 1 asserts SAI3 reset. Reading 1 means that SAI3 reset is asserted.

20.8.141 RCC SAI4 configuration register (RCC_SAI4CFGR)

Address offset: 0x7D0

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SAI4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4LPEN	SAI4EN	SAI4RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SAI4LPEN**: SAI4 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SAI4EN**: SAI4 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SAI4RST**: SAI4 reset

Set and cleared by software.

0: Writing 0 releases SAI4 reset. Reading 0 means that SAI4 reset is released.

1: Writing 1 asserts SAI4 reset. Reading 1 means that SAI4 reset is asserted.

20.8.142 RCC MDF1 configuration register (RCC_MDF1CFGR)

Address offset: 0x7D8

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the MDF1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDF1LPEN	MDF1EN	MDF1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **MDF1LPEN**: MDF1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **MDF1EN**: MDF1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **MDF1RST**: MDF1 reset
Set and cleared by software.
0: Writing 0 releases MDF1 reset. Reading 0 means that MDF1 reset is released.
1: Writing 1 asserts MDF1 reset. Reading 1 means that MDF1 reset is asserted.

20.8.143 RCC FDCAN configuration register (RCC_FDCANCFGR)

Address offset: 0x7E0

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the FDCAN and CANRAM.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN LPEN	FDCAN EN	FDCAN RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FDCANLPEN**: FDCAN clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **FDCANEN**: FDCAN clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **FDCANRST**: FDCAN reset
Set and cleared by software.
0: Writing 0 releases FDCAN reset. Reading 0 means that FDCAN reset is released.
1: Writing 1 asserts FDCAN reset. Reading 1 means that FDCAN reset is asserted.

20.8.144 RCC HDP configuration register (RCC_HDPCFGR)

Address offset: 0x7E4

Reset value: 0x0000 0000

This register is used to control reset and clock gating behavior of the HDP.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	HDPRST
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HDPEN**: HDP clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HDPRST**: HDP reset
Set and cleared by software.
0: Writing 0 releases HDP reset. Reading 0 means that HDP reset is released.
1: Writing 1 asserts HDP reset. Reading 1 means that HDP reset is asserted.

20.8.145 RCC ADC1 configuration register (RCC_ADC1CFGR)

Address offset: 0x7E8

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the ADC1.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADC1KERSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC1LPEN	ADC1EN	ADC1RST
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ADC1KERSEL**: ADC1 kernel clock source select
Set and cleared by software.
0x0: Kernel source clock is ck_ker_adc1.
0x1: Kernel source clock is ck_icn_ls_mcu.

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **ADC1LPEN**: ADC1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **ADC1EN**: ADC1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **ADC1RST**: ADC1 reset
Set and cleared by software.
0: Writing 0 releases ADC1 reset. Reading 0 means that ADC1 reset is released.
1: Writing 1 asserts ADC1 reset. Reading 1 means that ADC1 reset is asserted.

20.8.146 RCC ADC2 configuration register (RCC_ADC2CFGR)

Address offset: 0x7EC

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the ADC2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADC2KERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC2LPEN	ADC2EN	ADC2RST
		rw	rw										rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:12 **ADC2KERSEL[1:0]**: ADC2 kernel clock source select
Set and cleared by software.
0x0: Kernel source clock is ck_ker_adc2.
0x1: Kernel source clock is ck_icn_ls_mcu.
0x2: Kernel source clock is ck_ker_adc1.
0x3: Reserved

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **ADC2LPEN**: ADC2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **ADC2EN**: ADC2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **ADC2RST**: ADC2 reset
Set and cleared by software.
0: Writing 0 releases ADC2 reset. Reading 0 means that ADC2 reset is released.
1: Writing 1 asserts ADC2 reset. Reading 1 means that ADC2 reset is asserted.

20.8.147 RCC Ethernet 1 configuration register (RCC_ETH1CFGR)

Address offset: 0x7F0

Reset value: 0x0000 0A44

This register is used to control reset, clock gating, and low-power behavior of the Ethernet 1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1RXEN	ETH1TXLPEN	ETH1TXEN	Res.	ETH1LPEN	ETH1EN	ETH1STPEN	Res.	ETH1MACLPEN	ETH1MACEN	ETH1RST
				rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **ETH1RXLPEN**: Ethernet 1 reception clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables reception clock in CSleep. Reading 0 means that reception clock is disabled in CSleep.
1: Writing 1 enables reception clock in CSleep. Reading 1 means that reception clock is enabled in CSleep.

Bit 10 **ETH1RXEN**: Ethernet 1 reception clock enable
Set and cleared by software.
0: Writing 0 disables reception clock. Reading 0 means that reception clock is disabled.
1: Writing 1 enables reception clock. Reading 1 means that reception clock is enabled.

Bit 9 **ETH1TXLPEN**: Ethernet 1 transmission clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables transmission clock in CSleep. Reading 0 means that transmission clock is disabled in CSleep.
1: Writing 1 enables transmission clock in CSleep. Reading 1 means that transmission clock is enabled in CSleep.

Bit 8 **ETH1TXEN**: Ethernet 1 transmission clock enable
Set and cleared by software.
0: Writing 0 disables transmission clock. Reading 0 means that transmission clock is disabled.
1: Writing 1 enables transmission clock. Reading 1 means that transmission clock is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ETH1LPEN**: Ethernet 1 kernel clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables kernel clocks in CSleep. Reading 0 means that kernel clocks are disabled in CSleep.

1: Writing 1 enables kernel clocks in CSleep. Reading 1 means that kernel clocks are enabled in CSleep.

Bit 5 **ETH1EN**: Ethernet 1 kernel clock enable

Set and cleared by software.

0: Writing 0 disables kernel clocks. Reading 0 means that kernel clocks are disabled.

1: Writing 1 enables kernel clocks. Reading 1 means that kernel clocks are enabled.

Bit 4 **ETH1STPEN**: Ethernet 1 kernel clocks enable during CStop

Set and cleared by software. This bits allows ETH1 to receive clocks coming from ETH1_TX_CLK and ETH1_RX_CLK pins during CStop.

0: Writing 0 disables ETH1 TX and RX kernel clocks in CStop. Reading 0 means that ETH1 TX and RX kernel clocks are disabled in CStop.

1: Writing 1 enables ETH1 TX and RX kernel clocks in CStop. Reading 1 means that ETH1 TX and RX kernel clocks are enabled in CStop.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ETH1MACLPEN**: Ethernet 1 bus clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables bus clocks in CSleep. Reading 0 means that bus clocks are disabled in CSleep.

1: Writing 1 enables bus clocks in CSleep. Reading 1 means that bus clocks are enabled in CSleep.

Bit 1 **ETH1MACEN**: Ethernet 1 bus clock enable

Set and cleared by software.

0: Writing 0 disables bus clocks. Reading 0 means that bus clocks are disabled.

1: Writing 1 enables bus clocks. Reading 1 means that bus clocks are enabled.

Bit 0 **ETH1RST**: Ethernet 1 reset

Set and cleared by software.

0: Writing 0 releases ETH1 reset. Reading 0 means that ETH1 reset is released.

1: Writing 1 asserts ETH1 reset. Reading 1 means that ETH1 reset is asserted.

20.8.148 RCC Ethernet 2 configuration register (RCC_ETH2CFGR)

Address offset: 0x7F4

Reset value: 0x0000 0A44

This register is used to control reset, clock gating, and low-power behavior of the Ethernet 2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ETH2RXLPEN	ETH2RXEN	ETH2TXLPEN	ETH2TXEN	Res.	ETH2LPEN	ETH2EN	ETH2STPEN	Res.	ETH2MACLPEN	ETH2MACEN	ETH2RST
				rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **ETH2RXLPEN**: Ethernet 2 reception clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables reception clock in CSleep. Reading 0 means that reception clock is disabled in CSleep.

1: Writing 1 enables reception clock in CSleep. Reading 1 means that reception clock is enabled in CSleep.

Bit 10 **ETH2RXEN**: Ethernet 2 reception clock enable

Set and cleared by software.

0: Writing 0 disables reception clock. Reading 0 means that reception clock is disabled.

1: Writing 1 enables reception clock. Reading 1 means that reception clock is enabled.

Bit 9 **ETH2TXLPEN**: Ethernet 2 transmission clock enable during CSleep mode

Set and cleared by software.

0: Writing 0 disables transmission clock in CSleep. Reading 0 means that transmission clock is disabled in CSleep.

1: Writing 1 enables transmission clock in CSleep. Reading 1 means that transmission clock is enabled in CSleep.

Bit 8 **ETH2TXEN**: Ethernet 2 transmission clock enable

Set and cleared by software.

0: Writing 0 disables transmission clock. Reading 0 means that transmission clock is disabled.

1: Writing 1 enables transmission clock. Reading 1 means that transmission clock is enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ETH2LPEN**: Ethernet 2 kernel clocks enable during CSleep

Set and cleared by software.

0: Writing 0 disables kernel clocks in CSleep. Reading 0 means that kernel clocks are disabled in CSleep.

1: Writing 1 enables kernel clocks in CSleep. Reading 1 means that kernel clocks are enabled in CSleep.

Bit 5 **ETH2EN**: Ethernet 2 kernel clocks enable

Set and cleared by software.

0: Writing 0 disables kernel clocks. Reading 0 means that kernel clocks are disabled.

1: Writing 1 enables kernel clocks. Reading 1 means that kernel clocks are enabled.

Bit 4 **ETH2STPEN**: Ethernet 2 kernel clocks enable during CStop

Set and cleared by software. This bits allows ETH2 to receive clocks coming from ETH2_TX_CLK and ETH2_RX_CLK pins during CStop.

0: Writing 0 disables ETH2 TX and RX kernel clocks in CStop. Reading 0 means that ETH2 TX and RX kernel clocks are disabled in CStop.

1: Writing 1 enables ETH2 TX and RX kernel clocks in CStop. Reading 1 means that ETH2 TX and RX kernel clocks are enabled in CStop.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ETH2MACLPEN**: Ethernet 2 bus clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables bus clocks in CSleep. Reading 0 means that bus clocks are disabled in CSleep.

1: Writing 1 enables bus clocks in CSleep. Reading 1 means that bus clocks are enabled in CSleep.

Bit 1 **ETH2MACEN**: Ethernet 2 bus clock enable
Set and cleared by software.
0: Writing 0 disables bus clocks. Reading 0 means that bus clocks are disabled.
1: Writing 1 enables bus clocks. Reading 1 means that bus clocks are enabled.

Bit 0 **ETH2RST**: Ethernet 2 reset
Set and cleared by software.
0: Writing 0 releases ETH2 reset. Reading 0 means that ETH2 reset is released.
1: Writing 1 asserts ETH2 reset. Reading 1 means that ETH2 reset is asserted.

20.8.149 RCC USBH configuration register (RCC_USBHCFGR)

Address offset: 0x7FC

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USBH. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	USBHEN	USBHRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USBHLPEN**: USBH clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USBHEN**: USBH peripheral clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USBHRST**: USBH reset
Set and cleared by software.
0: Writing 0 releases USBH reset. Reading 0 means that USBH reset is released.
1: Writing 1 asserts USBH reset. Reading 1 means that USBH reset is asserted.

20.8.150 RCC USB2PHY1 configuration register (RCC_USB2PHY1CFGR)

Address offset: 0x800

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USB2PHY1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB2PHY1CKREFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1LPEN	USB2PHY1EN	USB2PHY1RST
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **USB2PHY1CKREFSEL**: USB2PHY1 reference clock selection

Set and cleared by software.

0: USB2PHY1 reference clock is channel 57 (ck_ker_usb2phy1) of the flexclkgen (default after reset).

1: USB2PHY1 reference clock is the HSE oscillator divided by 2 (hse_div2_ck).

Bits 14:3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY1LPEN**: USB2PHY1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USB2PHY1EN**: USB2PHY1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USB2PHY1RST**: USB2PHY1 reset

Set and cleared by software.

0: Writing 0 releases USB2PHY1 reset. Reading 0 means that USB2PHY1 reset is released.

1: Writing 1 asserts USB2PHY1 reset. Reading 1 means that USB2PHY1 reset is asserted.

20.8.151 RCC OTG configuration register (RCC_OTGCFGR)

Address offset: 0x808

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the OTG. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTGLPEN	OTGEN	OTGRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **OTGLPEN**: OTG clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **OTGEN**: OTG peripheral clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **OTGRST**: OTG reset

Set and cleared by software.

0: Writing 0 releases OTG reset. Reading 0 means that OTG reset is released.

1: Writing 1 asserts OTG reset. Reading 1 means that OTG reset is asserted.

20.8.152 RCC USB2PHY2 configuration register (RCC_USB2PHY2CFGR)

Address offset: 0x80C

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the USB2PHY2.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB2PHY2CKREFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2LPEN	USB2PHY2EN	USB2PHY2RST
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **USB2PHY2CKREFSEL**: USB2PHY2 reference clock select
Set and cleared by software.
0: USB2PHY2 reference clock is channel 58 (ck_ker_usb2phy2) of the flexclkgen (default after reset).
1: USB2PHY2 reference clock is the HSE oscillator divided by 2 (hse_div2_ck).

Bits 14:3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY2LPEN**: USB2PHY2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **USB2PHY2EN**: USB2PHY2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **USB2PHY2RST**: USB2PHY2 reset
Set and cleared by software.
0: Writing 0 releases USB2PHY2 reset. Reading 0 means that USB2PHY2 reset is released.
1: Writing 1 asserts USB2PHY2 reset. Reading 1 means that USB2PHY2 reset is asserted.

20.8.153 RCC STGEN configuration register (RCC_STGENCFGR)

Address offset: 0x824

Reset value: 0x0000 0004

This register is used to control clock gating and low-power clock mode behavior of the STGENC and STGENR.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGEN STPEN	Res.	STGEN LPEN	STGEN EN	Res.
											rw		rw	rw	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **STGENSTPEN**: STGENRW and STGENRO kernel clock enable during CStop
Set and cleared by software.
0: Writing 0 disables kernel clocks in CStop. Reading 0 means that kernel clocks are disabled in CStop.
1: Writing 1 enables kernel clocks in CStop. Reading 1 means that kernel clocks are enabled in CStop.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **STGENLPEN**: STGENRW and STGENRO clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **STGENEN**: STGENRW and STGENRO clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.154 RCC SDMMC1 configuration register (RCC_SDMMC1CFGR)

Address offset: 0x830

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SDMMC1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1DLLRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN	SDMMC1EN	SDMMC1RST
													rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC1DLLRST**: SDMMC1 DLL reset
Set and cleared by software.
0: Writing 0 releases DLL reset. Reading 0 means that DLL reset is released.
1: Writing 1 asserts DLL reset. Reading 1 means that DLL reset is asserted.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **SDMMC1LPEN**: SDMMC1 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SDMMC1EN**: SDMMC1 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SDMMC1RST**: SDMMC1 reset
Set and cleared by software.
0: Writing 0 releases SDMMC1 reset. Reading 0 means that SDMMC1 reset is released.
1: Writing 1 asserts SDMMC1 reset. Reading 1 means that SDMMC1 reset is asserted.

20.8.155 RCC SDMMC2 configuration register (RCC_SDMMC2CFGR)

Address offset: 0x834

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SDMMC2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2LPEN	SDMMC2EN	SDMMC2RST
													rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC2DLLRST**: SDMMC2 DLL reset
Set and cleared by software.
0: Writing 0 releases DLL reset. Reading 0 means that DLL reset is released.
1: Writing 1 asserts DLL reset. Reading 1 means that DLL reset is asserted.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **SDMMC2LPEN**: SDMMC2 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SDMMC2EN**: SDMMC2 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SDMMC2RST**: SDMMC2 reset
Set and cleared by software.
0: Writing 0 releases SDMMC2 reset. Reading 0 means that SDMMC2 reset is released.
1: Writing 1 asserts SDMMC2 reset. Reading 1 means that SDMMC2 reset is asserted.

20.8.156 RCC SDMMC3 configuration register (RCC_SDMMC3CFGR)

Address offset: 0x838

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the SDMMC3. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3DLLRST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN	SDMMC3EN	SDMMC3RST
													rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3DLLRST**: SDMMC3 DLL reset

Set and cleared by software.

0: Writing 0 releases DLL reset. Reading 0 means that DLL reset is released.

1: Writing 1 asserts DLL reset. Reading 1 means that DLL reset is asserted.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **SDMMC3LPEN**: SDMMC3 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SDMMC3EN**: SDMMC3 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SDMMC3RST**: SDMMC3 block reset

Set and cleared by software.

0: Writing 0 releases SDMMC3 reset. Reading 0 means that SDMMC3 reset is released.

1: Writing 1 asserts SDMMC3 reset. Reading 1 means that SDMMC3 reset is asserted.

20.8.157 RCC LTDC configuration register (RCC_LTDCCFGR)

Address offset: 0x840

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of LTDC.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN	LTDCEN	LTDCRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LTDCLPEN**: LTDC clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **LTDCEN**: LTDC peripheral clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **LTDCRST**: LTDC reset

Set and cleared by software.

0: Writing 0 releases LTDC reset. Reading 0 means that LTDC reset is released.

1: Writing 1 asserts LTDC reset. Reading 1 means that LTDC reset is asserted.

20.8.158 RCC CSI configuration register (RCC_CSICFGR)

Address offset: 0x858

Reset value: 0x0000 0004

This register is used to control reset, clock gating, and low-power behavior of the CSI.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSILPEN	CSIEN	CSIRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CSILPEN**: CSI clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **CSIEN**: CSI clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **CSIRST**: CSI reset

Set and cleared by software.

0: Writing 0 releases CSI reset. Reading 0 means that CSI reset is released.

1: Writing 1 asserts CSI reset. Reading 1 means that CSI reset is asserted.

20.8.159 RCC DCMIPP configuration register (RCC_DCMIPPCFGR)

Address offset: 0x85C

Reset value: 0x0000 0004

This register is used to control the reset, clock gating and low-power behavior of DCMIPP. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPPLPEN	DCMIPPEN	DCMIPPRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPLPEN**: DCMIPP clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DCMIPPEN**: DCMIPP clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DCMIPPRST**: DCMIPP reset

Set and cleared by software.

0: Writing 0 releases DCMIPP reset. Reading 0 means that DCMIPP reset is released.

1: Writing 1 asserts DCMIPP reset. Reading 1 means that DCMIPP reset is asserted.

20.8.160 RCC DCMIPSSI configuration register (RCC_DCMIPSSICFGR)

Address offset: 0x860

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of DCMIPSSI. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPSSILPEN	DCMIPSSIEN	DCMIPSSIRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DCMIPSSILPEN**: DCMIPSSI clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DCMIPSSIEN**: DCMIPSSI clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DCMIPSSIRST**: DCMIPSSI reset

Set and cleared by software.

0: Writing 0 releases DCMIPSSI reset. Reading 0 means that DCMIPSSI reset is released.

1: Writing 1 asserts DCMIPSSI reset. Reading 1 means that DCMIPSSI reset is asserted.

20.8.161 RCC RNG1 configuration register (RCC_RNG1CFGR)

Address offset: 0x870

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of RNG1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1LPEN	RNG1EN	RNG1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **RNG1LPEN**: RNG1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **RNG1EN**: RNG1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **RNG1RST**: RNG1 reset

Set and cleared by software.

0: Writing 0 releases RNG1 reset. Reading 0 means that RNG1 reset is released.

1: Writing 1 asserts RNG1 reset. Reading 1 means that RNG1 reset is asserted.

Note: The clocks of the following cryptographic peripherals must all be disabled before setting this bit: PKA, RNG1, SAES, HASH1, CRYP1.

20.8.162 RCC RNG2 configuration register (RCC_RNG2CFGR)

Address offset: 0x874

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of RNG2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG2LPEN	RNG2EN	RNG2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **RNG2LPEN**: RNG2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **RNG2EN**: RNG2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **RNG2RST**: RNG2 reset

Set and cleared by software.

0: Writing 0 releases RNG2 reset. Reading 0 means that RNG2 reset is released.

1: Writing 1 asserts RNG2 reset. Reading 1 means that RNG2 reset is asserted.

20.8.163 RCC PKA configuration register (RCC_PKACFGR)

Address offset: 0x878

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of PKA.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKALPEN	PKAEN	PKARST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **PKALPEN**: PKA clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **PKAEN**: PKA clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **PKARST**: PKA reset

Set and cleared by software.

0: Writing 0 releases PKA reset. Reading 0 means that PKA reset is released.

1: Writing 1 asserts PKA reset. Reading 1 means that PKA reset is asserted.

Note: The clocks of the following cryptographic peripherals must all be disabled before setting this bit: PKA, RNG1, SAES, HASH1, CRYP1.

20.8.164 RCC SAES configuration register (RCC_SAESCFGR)

Address offset: 0x87C

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of SAES.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESLPEN	SAESEN	SAESRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SAESLPEN**: SAES clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SAESEN**: SAES clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SAESRST**: SAES reset

Set and cleared by software.

0: Writing 0 releases SAES reset. Reading 0 means that SAES reset is released.

1: Writing 1 asserts SAES reset. Reading 1 means that SAES reset is asserted.

Note: The clocks of the following cryptographic peripherals must all be disabled before setting this bit: PKA, RNG1, SAES, HASH1, CRYP1.

20.8.165 RCC HASH1 configuration register (RCC_HASH1CFGR)

Address offset: 0x880

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of HASH1. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASH1 LPEN	HASH1 EN	HASH1 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HASH1LPEN**: HASH1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **HASH1EN**: HASH1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HASH1RST**: HASH1 reset

Set and cleared by software.

0: Writing 0 releases HASH1 reset. Reading 0 means that HASH1 reset is released.

1: Writing 1 asserts HASH1 reset. Reading 1 means that HASH1 reset is asserted.

Note: The clocks of the following cryptographic peripherals must all be disabled before setting this bit: PKA, RNG1, SAES, HASH1, CRYP1.

20.8.166 RCC HASH2 configuration register (RCC_HASH2CFGR)

Address offset: 0x884

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of HASH2. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASH2 LPEN	HASH2 EN	HASH2 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HASH2LPEN**: HASH2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **HASH2EN**: HASH2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **HASH2RST**: HASH2 reset

Set and cleared by software.

0: Writing 0 releases HASH2 reset. Reading 0 means that HASH2 reset is released.

1: Writing 1 asserts HASH2 reset. Reading 1 means that HASH2 reset is asserted.

20.8.167 RCC CRYP1 configuration register (RCC_CRYP1CFGR)

Address offset: 0x888

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of CRYP1.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYP1 LPEN	CRYP1 EN	CRYP1 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CRYP1LPEN**: CRYP1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **CRYP1EN**: CRYP1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **CRYP1RST**: CRYP1 reset

Set and cleared by software.

0: Writing 0 releases CRYP1 reset. Reading 0 means that CRYP1 reset is released.

1: Writing 1 asserts CRYP1 reset. Reading 1 means that CRYP1 reset is asserted.

Note: The clocks of the following cryptographic peripherals must all be disabled before setting this bit: PKA, RNG1, SAES, HASH1, CRYP1.

20.8.168 RCC CRYP2 configuration register (RCC_CRYP2CFGR)

Address offset: 0x88C

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of CRYP2.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYP2 LPEN	CRYP2 EN	CRYP2 RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CRYP2LPEN**: CRYP2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **CRYP2EN**: CRYP2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **CRYP2RST**: CRYP2 reset

Set and cleared by software.

0: Writing 0 releases CRYP2 reset. Reading 0 means that CRYP2 reset is released.

1: Writing 1 asserts CRYP2 reset. Reading 1 means that CRYP2 reset is asserted.

20.8.169 RCC IWDG1 configuration register (RCC_IWDG1CFGR)

Address offset: 0x894

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of IWDG1.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG1 LPEN	IWDG1 EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IWDG1LPEN**: IWDG1 bus clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.

1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clock is enabled in CSleep.

Bit 1 **IWDG1EN**: IWDG1 bus clock enable

Set and cleared by software.

0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.

1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.170 RCC IWDG2 configuration register (RCC_IWDG2CFGR)

Address offset: 0x898

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of IWDG2.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2 LPEN	IWDG2 EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IWDG2LPEN**: IWDG2 bus clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.
1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clock is enabled in CSleep.

Bit 1 **IWDG2EN**: IWDG2 bus clock enable
Set and cleared by software.
0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.
1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.171 RCC IWDG3 configuration register (RCC_IWDG3CFGR)

Address offset: 0x89C

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of IWDG3.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG3 LPEN	IWDG3 EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IWDG3LPEN**: IWDG3 bus clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.

1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clock is enabled in CSleep.

Bit 1 **IWDG3EN**: IWDG3 bus clock enable

Set and cleared by software.

0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.

1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.172 RCC IWDG4 configuration register (RCC_IWDG4CFGR)

Address offset: 0x8A0

Reset value: 0x0000 0004

This register is used to control clock gating and low-power behavior of IWDG4. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4 LPEN	IWDG4 EN	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IWDG4LPEN**: IWDG4 bus clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.

1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clock is enabled in CSleep.

Bit 1 **IWDG4EN**: IWDG4 bus clock enable

Set and cleared by software.

0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.

1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 Reserved, must be kept at reset value.

20.8.173 RCC WWDG1 configuration register (RCC_WWDG1CFGR)

Address offset: 0x8A4

Reset value: 0x0000 0004

This register is used to control th clock gating and low-power behavior of WWDG1.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG 1LPEN	WWDG 1EN	WWDG 1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WWDG1LPEN**: WWDG1 bus clock enable during CSleep
Set and cleared by software.
0: ‘Writing 0 disables APB clock in CSleep. Reading 0 means that APB clock is disabled in CSleep.
1: Writing 1 enables APB clock in CSleep. Reading 1 means that APB clock is enabled in CSleep.

Bit 1 **WWDG1EN**: WWDG1 bus clock enable
Set and cleared by software.
0: Writing 0 disables APB clock. Reading 0 means that APB clock is disabled.
1: Writing 1 enables APB clock. Reading 1 means that APB clock is enabled.

Bit 0 **WWDG1RST**: WWDG1 reset
Set and cleared by software.
0: Writing 0 releases WWDG1 reset. Reading 0 means that WWDG1 reset is released.
1: Writing 1 asserts WWDG1 reset. Reading 1 means that WWDG1 reset is asserted.

20.8.174 RCC VREFBUF configuration register (RCC_VREFCFGR)

Address offset: 0x8AC

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of VREFBUF.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFL PEN	VREFE N	VREFR ST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **VREFLPEN**: VREFBUF clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **VREFEN**: VREFBUF clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **VREFRST**: VREFBUF reset
Set and cleared by software.
0: Writing 0 releases VREFBUF reset. Reading 0 means that VREFBUF reset is released.
1: Writing 1 asserts VREFBUF reset. Reading 1 means that VREFBUF reset is asserted

20.8.175 RCC DTS configuration register (RCC_DTSCFGR)

Address offset: 0x8B0

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of DTS.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTSKERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	DTSEN	DTSRST
		rw	rw										rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:12 **DTSKERSEL[1:0]**: DTS kernel clock selection
Set and cleared by software.
0x0: hsi_ker_ck selected
0x1: hse_ker_ck selected
0x2: msi_ker_ck selected
0x3: Reserved

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DTSLPEN**: DTS clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DTSEN**: DTS clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DTSRST**: DTS reset
Set and cleared by software.
0: Writing 0 releases DTS reset. Reading 0 means that DTS reset is released.
1: Writing 1 asserts DTS reset. Reading 1 means that DTS reset is asserted

20.8.176 RCC CRC configuration register (RCC_CRCCFGR)

Address offset: 0x8B4

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of CRC.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	CRCEN	CRCRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CRCLPEN**: CRC clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **CRCEN**: CRC clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **CRCRST**: CRC reset
Set and cleared by software.
0: Writing 0 releases CRC reset. Reading 0 means that CRC reset is released.
1: Writing 1 asserts CRC reset. Reading 1 means that CRC reset is asserted

20.8.177 RCC SERC configuration register (RCC_SERCCFGR)

Address offset: 0x8B8

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of SERC.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SERCLPEN	SERCEN	SERCRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SERCLPEN**: SERC clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **SERCEN**: SERC clocks enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **SERCRST**: SERC reset

Set and cleared by software.

0: Writing 0 releases SERC reset. Reading 0 means that SERC reset is released.

1: Writing 1 asserts SERC reset. Reading 1 means that SERC reset is asserted

20.8.178 RCC DDRPERFM configuration register (RCC_DDRPERFMCFGR)

Address offset: 0x8C0

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of DDRPERFM.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	DDRPERFMEN	DDRPERFMRST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DDRPERFMLPEN**: DDRPERFM clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **DDRPERFMEN**: DDRPERFM clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **DDRPERFMRST**: DDRPERFM reset

Set and cleared by software.

0: Writing 0 releases DDRPERFM reset. Reading 0 means that DDRPERFM reset is released.

1: Writing 1 asserts DDRPERFM reset. Reading 1 means that DDRPERFM reset is asserted.

20.8.179 RCC I3C1 configuration register (RCC_I3C1CFGR)

Address offset: 0x8C8

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of I3C1.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3C1LPEN	I3C1EN	I3C1RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I3C1LPEN**: I3C1 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I3C1EN**: I3C1 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I3C1RST**: I3C1 reset

Set and cleared by software.

0: Writing 0 releases I3C1 reset. Reading 0 means that I3C1 reset is released.

1: Writing 1 asserts I3C1 reset. Reading 1 means that I3C1 reset is asserted.

20.8.180 RCC I3C2 configuration register (RCC_I3C2CFGR)

Address offset: 0x8CC

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of I3C2.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3C2LPEN	I3C2EN	I3C2RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I3C2LPEN**: I3C2 clock enable during CSleep

Set and cleared by software.

0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.

1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I3C2EN**: I3C2 clock enable

Set and cleared by software.

0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.

1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I3C2RST**: I3C2 reset

Set and cleared by software.

0: Writing 0 releases I3C2 reset. Reading 0 means that I3C2 reset is released.

1: Writing 1 asserts I3C2 reset. Reading 1 means that I3C2 reset is asserted.

20.8.181 RCC I3C3 configuration register (RCC_I3C3CFGR)

Address offset: 0x8D0

Reset value: 0x0000 0004

This register is used to control reset, clock gating and low-power behavior of I3C3.
Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3C3LPEN	I3C3EN	I3C3RST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **I3C3LPEN**: I3C3 clock enable during CSleep
Set and cleared by software.
0: Writing 0 disables peripheral clocks in CSleep. Reading 0 means that peripheral clocks are disabled in CSleep.
1: Writing 1 enables peripheral clocks in CSleep. Reading 1 means that peripheral clocks are enabled in CSleep.

Bit 1 **I3C3EN**: I3C3 clock enable
Set and cleared by software.
0: Writing 0 disables peripheral clocks. Reading 0 means that peripheral clocks are disabled.
1: Writing 1 enables peripheral clocks. Reading 1 means that peripheral clocks are enabled.

Bit 0 **I3C3RST**: I3C3 reset
Set and cleared by software.
0: Writing 0 releases I3C3 reset. Reading 0 means that I3C3 reset is released.
1: Writing 1 asserts I3C3 reset. Reading 1 means that I3C3 reset is asserted.

20.8.182 RCC MUXSEL configuration register (RCC_MUXSELCFGR)

Address offset: 0x1000

Reset value: 0x0000 0000

This register is used to select reference clock for each channel in MUXSEL of flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MUXSEL6[1:0]		Res.	Res.	MUXSEL5[1:0]		Res.	MUXSEL4[2:0]		
						rw	rw			rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MUXSEL3[2:0]			Res.	MUXSEL2[2:0]			Res.	MUXSEL1[2:0]			Res.	MUXSEL0[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **MUXSEL6[1:0]**: PLL2 reference clock selection (ck_pll2_ref)
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x3: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **MUXSEL5[1:0]**: PLL1 reference clock selection (ck_pll1_ref)
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x3: Reserved

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **MUXSEL4[2:0]**: PLL8 reference clock selection
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x5: ck_in
Others: Reserved

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MUXSEL3[2:0]**: PLL7 reference clock selection
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x5: ck_in
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **MUXSEL2[2:0]**: PLL6 reference clock selection
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x5: ck_in
Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **MUXSEL1[2:0]**: PLL5 reference clock selection
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x5: ck_in
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **MUXSEL0[2:0]**: PLL4 reference clock selection
Set and cleared by software.
0x0: hsi_ck
0x1: hse_ck
0x2: msi_ck
0x5: ck_in
Others: Reserved

20.8.183 RCC cross bar x configuration register (RCC_XBARxCFGR)

Address offset: 0x1018 + 0x4 * x, (x = 0 to 63)

Reset value: 0x0000 0045

This register is used to configure channel x output of cross bar (XBAR) in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number x configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XBARx STS	XBARx EN	Res.	Res.	XBARxSEL[3:0]			
								r	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **XBARxSTS**: Cross bar channel x output status
Set and cleared by hardware to indicate channel x output status versus setting changes.
0: Channel x is aligned to given settings.
1: Indicates change ongoing.

Bit 6 **XBARxEN**: Cross bar channel x clock output enable
Set and cleared by software. It is recommended to keep the channel x enabled at cross bar stage, and to use the final divider enable bit to disable the channel x output.
0: Channel x output disabled
1: Channel x output enabled

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **XBARxSEL[3:0]**: Cross bar channel x input clock selection
Set and cleared by software to select the input clock for the channel x.
0x0: PLL4 FOUTPOSTDIV clock selected
0x1: PLL5 FOUTPOSTDIV clock selected
0x2: PLL6 FOUTPOSTDIV clock selected
0x3: PLL7 FOUTPOSTDIV clock selected
0x4: PLL8 FOUTPOSTDIV clock selected
0x5: hsi_ck selected
0x6: hse_ck selected
0x7: msi_ck selected
0x8: hsi_ker_ck selected
0x9: hse_ker_ck selected
0xA: msi_ker_ck selected
0xB: spdifrx_symb_ck selected
0xC: ck_in selected
0xD: lsi_ck selected
0xE: lse_ck selected
Other: Reserved

20.8.184 RCC pre divider x configuration register (RCC_PREDIVxCFGR)

Address offset: 0x1118 + 0x4 * x, (x = 0 to 63)

Reset value: 0x0000 0000

This register is used to configure the channel x predivider (PREDIV) stage in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number x configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PREDIVx[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **PREDIVx[9:0]**: Predivider channel x divider ratio

Set and cleared by software to select one div1, div2, div4 or div1024 ratio.

0x0: div1

0x1: div2

0x3: div4

0x3FF: div1024

Others: Reserved

20.8.185 RCC pre divider status register 1 (RCC_PREDIVSR1)

Address offset: 0x1218

Reset value: 0x0000 0000

This register is used to get status of first 32 channel outputs of PREDIV in flexclkgen. Word format data access only supported on this register. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PREDIVSTS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREDIVSTS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PREDIVSTS[31:0]**: Predivider channel[31:0] output status

Set by hardware to indicate the status of channel output versus divide ratio changes.

0: Channel is aligned with divide ratio change.

1: Indicates that the divider is in the process of changing.

20.8.186 RCC pre divider status register 2 (RCC_PREDIVSR2)

Address offset: 0x121C

Reset value: 0x0000 0000

This register is used to get status of last 32 channel outputs of PREDIV in flexclkgen. Word format data access only supported on this register. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PREDIVSTS[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREDIVSTS[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PREDIVSTS[63:32]**: Predivider channel[63:32] output status

Set by hardware to indicate the status of channel output versus divide ratio changes.

0: Channel is aligned with divide ratio change.

1: Indicates that the divider is in the process of changing.

20.8.187 RCC final divider x configuration register (RCC_FINDIVxCFGR)

Address offset: 0x1224 + 0x4 * x, (x = 0 to 63)

Reset value: 0x0000 0040

This register is used to configure channel x final divider (FINDIV) in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number x configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINDIVxEN	FINDIVx[5:0]					
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **FINDIVxEN**: Final divider channel x output enable

Set and cleared by software.

0: Channel x output disabled

1: Channel x output enabled

Bits 5:0 **FINDIVx[5:0]**: Channel x divider ratio

Set and cleared by software to select one div1 up to div64 step.

20.8.188 RCC final divider status register 1 (RCC_FINDIVSR1)

Address offset: 0x1324

Reset value: 0x0000 0000

This register is used to get status of first 32 channel outputs of FINDIV in flexclkgen. Word format access only supported on this register. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FINDIVSTS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FINDIVSTS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **FINDIVSTS[31:0]**: Final divider channel[31:0] output status

Set by hardware to indicate the status of the channel output vs divide ratio changes.

0: Channel is aligned with divide ratio change.

1: Indicates that the divider is in the process of changing.

20.8.189 RCC final divider status register 2 (RCC_FINDIVSR2)

Address offset: 0x1328

Reset value: 0x0000 0000

This register is used to get status of last 32 channel outputs of FINDIV in flexclkgen. Word format access only supported on this register. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FINDIVSTS[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FINDIVSTS[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **FINDIVSTS[63:32]**: Final divider channel[63:32] output status

Set by hardware to indicate the status of the channel output versus divide ratio changes.

0: Channel is aligned with divide ratio change.

1: Indicates that the divider is in the process of changing.

20.8.190 RCC clock frequency calculator and observation 0 clock configuration register (RCC_FCALCOBS0CFGR)

Address offset: 0x1340

Reset value: 0x1600 0087

This register is used to configure clock frequency calculator and observation clock 0 in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 65 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CKOBSEN	FCALCCKEN	CKOBSDIV[2:0]			Res.	Res.	Res.	CKOBSINV	FCALCCKINV	CKOBSEXTSEL
					rw	rw	rw	rw	rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCALCCKEXTSEL	Res.	Res.	Res.	Res.	CKEXTSEL[2:0]			CKINTSEL[7:0]							
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **CKOBSEN**: Observation clock 0 output enable
Set and cleared by software to enable the observation clock 0 output in front of the divider.
0: Observation clock 0 output disabled
1: Observation clock 0 output enabled (default after reset)

Bit 25 **FCALCCKEN**: Clock frequency calculator input clock enable
Set and cleared by software to enable the clock frequency calculator input clock.
0: Input clock disabled
1: Input clock enabled (default after reset)

Bits 24:22 **CKOBSDIV[2:0]**: Observation clock 0 divide ratio
Set and cleared by software to select the divide ratio for the observation clock 0.
000: div1 (default after reset)
001: div2
010: div4
011: div8
100: div16
101: div32
110: div64
111: div128

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **CKOBSINV**: Observation clock 0 inverter
Set and cleared by software to invert the observation clock 0 in front of the divider.
0: Observation clock 0 not inverted (default after reset)
1: Observation clock 0 inverted

Bit 17 **FCALCCKINV**: Clock frequency calculator input clock inverter

Set and cleared by software to invert THE clock frequency calculator input clock (in front of the divider).

0: Input clock not inverted (default after reset)

1: Input clock inverted

Bit 16 **CKOBSEXTSEL**: External clock source for observation clock 0

Set and cleared by software to select internal or external clocks for observation clock 0.

0: Internal clock selected for observation clock 0 (default after reset)

1: External clock selected for observation clock 0

Bit 15 **FCALCCKEXTSEL**: External clock source for clock frequency calculator input clock

Set and cleared by software to select an internal or external clock source for the clock frequency calculator input clock.

0: Internal clock selected for the clock frequency calculator input clock (default after reset)

1: External clock selected for the clock frequency calculator input clock

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **CKEXTSEL[2:0]**: External clock selector for clock frequency calculator and observation clock 0

Set and cleared by software to select the external clock to be calculated or observed.

000: pll1 clock divided by 42

001: pll2 clock divided by 8

010: Reserved

011: usb2phy1 PLL clock

100: usb2phy2 PLL clock

101: Reserved

110: Reserved

111: Reserved

Bits 7:0 **CKINTSEL[7:0]**: Internal clock selector for clock frequency calculator and observation clock 0
Set and cleared by software to select the internal clock to be calculated or observed.
0000 0000: hsi_ck
0000 0001: hse_ck
0000 0010: msi_ck
0000 0110: PLL4 reference input clock
0000 0111: PLL5 reference input clock
0000 1000: PLL6 reference input clock
0000 1001: PLL7 reference input clock
0000 1010: PLL8 reference input clock
0000 1011: PLL1 reference input clock
0000 1100: PLL2 reference input clock
0100 0000: PLL4 output clock
0100 0001: PLL5 output clock
0100 0010: PLL6 output clock
0100 0011: PLL7 output clock
0100 0100: PLL8 output clock
1000 0000: hsi_ker_ck
1000 0001: hse_ker_ck
1000 0010: msi_ker_ck
1000 0011: spdifrx_symb_ck
1000 0100: ck_in
1000 0101: lsi_ck
1000 0110: lse_ck
1000 0111: ck_xbar_fsm (default after reset)
1000 1000: ck_icn_p_rcc
1100 0000: findiv[0]
1100 0001: findiv[1]
...
1111 1111: findiv[63]
Other: Reserved

20.8.191 RCC clock frequency calculator and observation 1 clock configuration register (RCC_FCALCOBS1CFGR)

Address offset: 0x1344

Reset value: 0x0600 0087

This register is used to configure clock frequency calculator and observation clock 1 in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 65 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FCALCRSTN	CKOBSEN	Res.	CKOBSDIV[2:0]			Res.	Res.	Res.	CKOBSINV	Res.	CKOBSEXTSEL
				rw	rw		rw	rw	rw				rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CKEXTSEL[2:0]			CKINTSEL[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **FCALCRSTN**: Clock frequency calculator reset
Set and cleared by software.
0: Clock frequency calculator reset module asserted (default after reset)
1: Clock frequency calculator reset module released

Bit 26 **CKOBSEN**: Observation clock 1 output enable
Set and cleared by software to enable observation clock 1 output in front of the divider.
0: Observation clock 1 output disabled
1: Observation clock 1 output enabled (default after reset)

Bit 25 Reserved, must be kept at reset value.

Bits 24:22 **CKOBSDIV[2:0]**: Observation clock 1 divide ratio
Set and cleared by software to select the divide ratio for observation clock 1.
000: div1 (default after reset)
001: div2
010: div4
011: div8
100: div16
101: div32
110: div64
111: div128

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **CKOBSINV**: Observation clock 1 inverter
Set and cleared by software to invert observation clock 1 in front of the divider.
0: Observation clock 1 not inverted (default after reset)
1: Observation clock 1 inverted

Bit 17 Reserved, must be kept at reset value.

Bit 16 **CKOBSEXTSEL**: External clock source for the observation clock 1
Set and cleared by software to select between internal or external clocks for observation clock 1.
0: Internal clock selected for observation clock 1(default after reset)
1: External clock selected for observation clock 1

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **CKEXTSEL[2:0]**: External clock for observation clock 1
Set and cleared by software to select the external clock to be observed.
000: pll1 clock divided by 42
001: pll2 clock divided by 8
010: Reserved
011: usb2phy1 PLL clock
100: usb2phy2 PLL clock
101: Reserved
110: Reserved
111: Reserved

Bits 7:0 **CKINTSEL[7:0]**: Internal clock selector for observation clock 1

Set and cleared by software to select the internal clock to be observed.

0000 0000: hsi_ck
0000 0001: hse_ck
0000 0010: msi_ck
0000 0110: PLL4 reference input clock
0000 0111: PLL5 reference input clock
0000 1000: PLL6 reference input clock
0000 1001: PLL7 reference input clock
0000 1010: PLL8 reference input clock
0000 1011: PLL1 reference input clock
0000 1100: PLL2 reference input clock
0100 0000: PLL4 output clock
0100 0001: PLL5 output clock
0100 0010: PLL6 output clock
0100 0011: PLL7 output clock
0100 0100: PLL8 output clock
1000 0000: hsi_ker_ck
1000 0001: hse_ker_ck
1000 0010: msi_ker_ck
1000 0011: spdifrx_symb_ck
1000 0100: ck_in
1000 0101: lsi_ck
1000 0110: lse_ck
1000 0111: ck_xbar_fsm (default after reset)
1000 1000: ck_icn_p_rcc
1100 0000: findiv[0]
1100 0001: findiv[1]
...
1111 1111: findiv[63]
Others: Reserved

20.8.192 RCC clock frequency calculator reference clock configuration register (RCC_FCALCREFCFGR)

Address offset: 0x1348

Reset value: 0x0000 4000

This register is used to configure clock frequency calculator reference clock in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 65 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCREFCKSEL[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **FCALCREFCKSEL[2:0]**: Reference clock selection used by clock frequency calculator

Set and cleared by software to select clock frequency calculator reference clock used for definition of clock frequency calculation time window (see RCC_FCALCCR2).

000: PLL4 reference input clock
001: PLL5 reference input clock
010: PLL6 reference input clock
011: PLL7 reference input clock
100: PLL8 reference input clock
Others: Reserved

20.8.193 RCC clock frequency calculator control register 1 (RCC_FCALCCR1)

Address offset: 0x134C

Reset value: 0x0000 0000

This register is used to control clock frequency calculator in flexclkgen block. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 65 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALC RUN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FCALCRUN**: Clock frequency calculator run

Set and cleared by software to run the measure.

0: Clock frequency calculator off (default after reset)
1: Clock frequency calculator on

20.8.194 RCC clock frequency calculator control register 2 (RCC_FCALCCR2)

Address offset: 0x1354

Reset value: 0x0000 0000

This register is used to control the clock frequency calculator in flexclkgen. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 65 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCTYP[4:0]					Res.
										rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FCALCTWC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	FCALCMD[1:0]		Res.	Res.	Res.
	rw	rw	rw	rw							rw	rw			

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:17 **FCALCTYP[4:0]**: Clock frequency calculation type selection
Set and cleared by software to select the clock frequency calculation type.
0x0C: Frequency value type
Other: Reserved

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:11 **FCALCTWC[3:0]**: Time window code for clock frequency calculation
Set and cleared by software to set the number of reference clock cycles used for the clock frequency calculation in application mode (see FCALCMD).
0x0: 63 cycles
0x1: 3 cycles
0x2: 7 cycles
0x3: 15 cycles
0x4: 31 cycles
0x5: 63 cycles
0x6: 127 cycles
0x7: 255 cycles
0x8: 511 cycles
0x9: 1023 cycles
0xA: 2047 cycles
0xB: 4095 cycles
0xC: 8191 cycles
0xD: 16383 cycles
0xE: 32767 cycles
0xF: 65535 cycles

Bits 10:5 Reserved, must be kept at reset value.

Bits 4:3 **FCALCMD[1:0]**: Clock frequency calculation method
Set and cleared by software to set the clock frequency calculation method.
11: Application method
Other: Reserved

Bits 2:0 Reserved, must be kept at reset value.

20.8.195 RCC clock frequency calculator status register (RCC_FCALCSR)

Address offset: 0x1358

Reset value: 0x0000 0000

This register is used to get status of clock frequency calculator in flexclkgen. Word format data access only supported on this register. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCSTS	Res.	Res.	FVAL[16]
												r			r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FVAL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **FCALCSTS**: Clock frequency calculation status

Set by hardware to indicate the status of the clock frequency calculation.

– When FCALCRUN = 0, FCALCSTS = FVAL[16] (overflow of the register).

– When FCALCRUN = 1, FCALCSTS = 1 when the clock frequency calculation is ended.

Bits 18:17 Reserved, must be kept at reset value.

Bits 16:0 **FVAL[16:0]**: Clock frequency calculation value

Set by hardware to indicate the clock frequency calculation value.

– FVAL[16] = overflow flag of the frequency counter.

– When FCALCTYP = 0x0C, FVAL[15:0] = frequency counter value.

20.8.196 RCC PLLy configuration register 1 (RCC_PLLyCFGR1)

Address offset: 0x1360 + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x1000 0001

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CKREFST	Res.	Res.	Res.	PLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rc_w				r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSMODRST
							rw								rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CKREFST**: PLLy reference clock state
Set by hardware and cleared by software. Indicates if PLLy reference clock is running.
After any reset, this bit is updated after 5 clock cycles of the selected PLL reference clock.
0: Reading 0 means that PLLy reference clock does not run. Writing 0 resets the reference clock running detector.
1: Reading 1 means that PLLy reference clock is running. Writing 1 resets the reference clock running detector.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **PLLRDY**: PLLy clock ready flag
Set by hardware to indicate when PLLy is locked.
0: PLLy unlocked (default after reset)
1: PLLy locked

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **PLLEN**: PLLy enable
Set and cleared by software to enable PLLy.
0: PLLy off (default after reset)
1: PLLy on

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SSMODRST**: PLLy spread-spectrum modulator reset
Set and cleared by software.
0: PLLy spread-spectrum modulator reset module released
1: PLLy spread-spectrum modulator reset module asserted (default after reset)

20.8.197 RCC PLLy configuration register 2 (RCC_PLLyCFGR2)

Address offset: 0x1364 + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0000 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FBDIV[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **FBDIV[11:0]**: PLLy VCO multiplication factor
Set and cleared by software to control VCO multiplication factor.
In integer mode, this value must be set between 16 and 640.
In fractional mode, this value must be set between 20 and 320.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **FREFDIV[5:0]**: PLLy reference input clock divide frequency ratio
Set and cleared by software.
0x0: Not applicable
0x1: FREF divided by 1 (min value)
0x2: FREF divided by 2
0x3: FREF divided by 3
...
0x3C: FREF divided by 60
0x3D: FREF divided by 61
0x3E: FREF divided by 62
0x3F: FREF divided by 63

20.8.198 RCC PLLy configuration register 3 (RCC_PLLyCFGR3)

Address offset: 0x1368 + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0400 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SSCGDIS	DACEN	DOWNSPREAD	FRACIN[23:16]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SSCGDIS**: PLLy spread-spectrum modulator disable
Set and cleared by software to enable the PLLy clock spreading generator.
0: PLLy spread-spectrum modulator enabled
1: PLLy spread-spectrum modulator disabled (default after reset)

Bit 25 **DACEN**: PLLy noise canceling DAC enable in fractional mode.
Set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 24 **DOWNSPREAD**: PLLy VCO frequency modulation mode
Set and cleared by software to select the PLLy clock spreading mode.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bits 23:0 **FRACIN[23:0]**: Fractional part of PLLy VCO multiplication factor
Set and cleared by software to control the VCO fractional multiplication factor.
VCO output frequency = FREF * (FBDIV + FRAC / 2^{24}) / FREFDIV

20.8.199 RCC PLLy configuration register 4 (RCC_PLLyCFGR4)

Address offset: 0x136C + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0000 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BYPASS	FOUTPOSTDIVEN	DSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw								

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **BYPASS**: PLLy FOUTPOSTDIV bypass
Set and cleared by software to bypass the FOUTPOSTDIV by FREF.
0: FOUTPOSTDIV driven by POSTDIV2
1: FOUTPOSTDIV driven by FREF

Bit 9 **FOUTPOSTDIVEN**: PLLy output and post dividers enable
Set and cleared by software to enable the PLLy output and post dividers.
0: FOUTPOSTDIV output, POSTDIV1, and POSTDIV2 are powered down.
1: FOUTPOSTDIV output, POSTDIV1, and POSTDIV2 dividers are active.

Bit 8 **DSMEN**: PLLy delta-sigma modulator enable
Set and cleared by software to enable the delta-sigma modulator (DSM).
0: DSM powered down
1: DSM active

Bits 7:0 Reserved, must be kept at reset value.

20.8.200 RCC PLLy configuration register 5 (RCC_PLLyCFGR5)

Address offset: 0x1370 + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0000 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPREAD[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVVAL[3:0]			
												rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **SPREAD[4:0]**: Modulation depth adjustment for PLLy
Set and cleared by software to adjust modulation depth of the clock spreading generator.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **DIVVAL[3:0]**: Modulation frequency adjustment for PLLy
Set and cleared by software to adjust modulation frequency of the clock spreading generator.

20.8.201 RCC PLLy configuration register 6 (RCC_PLLyCFGR6)

Address offset: 0x1378 + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0000 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV1[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **POSTDIV1[2:0]**: PLLy VCO frequency divide level 1
Set and cleared by software to divide the VCO frequency output.
0x0: Not applicable
0x1: FOUTVCO divided by 1 (minimum value)
0x2: FOUTVCO divided by 2
0x3: FOUTVCO divided by 3
0x4: FOUTVCO divided by 4
0x5: FOUTVCO divided by 5
0x6: FOUTVCO divided by 6
0x7: FOUTVCO divided by 7

20.8.202 RCC PLLy configuration register 7 (RCC_PLLyCFGR7)

Address offset: 0x137C + 0x28 * (y - 4), (y = 4 to 8)

Reset value: 0x0000 0000

This register is used to configure flexclkgen PLLy. Word format data access only supported on this register.

Write accesses are defined according to the RCC local resource number 64 configuration (see *Table 161* for details). Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **POSTDIV2[2:0]**: PLLy VCO frequency divide level 2
Set and cleared by software to divide the VCO frequency output.
0x0: Not applicable
0x1: FOUTVCO/POSTDIV1 divided by 1 (minimum value)
0x2: FOUTVCO/POSTDIV1 divided by 2
0x3: FOUTVCO/POSTDIV1 divided by 3
0x4: FOUTVCO/POSTDIV1 divided by 4
0x5: FOUTVCO/POSTDIV1 divided by 5
0x6: FOUTVCO/POSTDIV1 divided by 6
0x7: FOUTVCO/POSTDIV1 divided by 7

Note: FOUTPOSTDIV = FOUTVCO / POSTDIV1 × POSTDIV2.

20.8.203 RCC version register (RCC_VERR)

Address offset: 0xFFF4

Reset value: 0x0000 0010

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major RCC revision

Bits 3:0 **MINREV[3:0]**: Minor RCC revision

20.8.204 RCC identifier register (RCC_IDR)

Address offset: 0xFFF8

Reset value: 0x0000 0003

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RCC identifier

20.8.205 RCC size identification register (RCC_SIDR)

Address offset: 0xFFFC

Reset value: 0xA3C5 DD40

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RCC decoding space (64 Kbytes)

20.8.206 RCC register map

Table 162. RCC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**RCC_SECCFGR0**	SEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**RCC_SECCFGR1**	SEC[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**RCC_SECCFGR2**	SEC95	SEC94	SEC93	SEC92	SEC91	SEC90	Res.	SEC88	SEC87	SEC86	SEC85	SEC84	SEC83	Res.	Res.	Res.	SEC79	SEC78	SEC77	SEC76	SEC75	SEC74	SEC73	SEC72	SEC71	SEC70	SEC69	SEC68	SEC67	SEC66	SEC65	SEC64
	Reset value	0	0	0	0	0	0		0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**RCC_SECCFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC113	SEC112	SEC111	SEC110	SEC109	SEC108	Res.	SEC106	SEC105	SEC104	SEC103	SEC102	SEC101	Res.	Res.	SEC98	SEC97	SEC96
	Reset value															0	0	0	0	0	0		0	0	0	0	0	0			0	0	0
0x010	**RCC_PRIVCFGR0**	PRIV[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**RCC_PRIVCFGR1**	PRIV[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**RCC_PRIVCFGR2**	PRIV95	PRIV94	PRIV93	PRIV92	PRIV91	PRIV90	Res.	PRIV88	PRIV87	PRIV86	PRIV85	PRIV84	PRIV83	Res.	Res.	Res.	PRIV79	PRIV78	PRIV77	PRIV76	PRIV75	PRIV74	PRIV73	PRIV72	PRIV71	PRIV70	PRIV69	PRIV68	PRIV67	PRIV66	PRIV65	PRIV64
	Reset value	0	0	0	0	0	0		0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**RCC_PRIVCFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV113	PRIV112	PRIV111	PRIV110	PRIV109	PRIV108	Res.	PRIV106	PRIV105	PRIV104	PRIV103	PRIV102	PRIV101	Res.	Res.	PRIV98	PRIV97	PRIV96
	Reset value															0	0	0	0	0	0		0	0	0	0	0	0			0	0	0
0x020	**RCC_RCFGLOCKR0**	RLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**RCC_RCFGLOCKR1**	RLOCK[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**RCC_RCFGLOCKR2**	RLOCK95	RLOCK94	RLOCK93	RLOCK92	RLOCK91	RLOCK90	Res.	RLOCK88	RLOCK87	RLOCK86	RLOCK85	RLOCK84	RLOCK83	Res.	Res.	Res.	RLOCK79	RLOCK78	RLOCK77	RLOCK76	RLOCK75	RLOCK74	RLOCK73	Res.	RLOCK71	RLOCK70	RLOCK69	RLOCK68	RLOCK67	RLOCK66	RLOCK65	RLOCK64
	Reset value	0	0	0	0	0	0		0	0	0	0	0	0				0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x02C	**RCC_RCFGLOCKR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLOCK113	RLOCK112	RLOCK111	RLOCK110	RLOCK109	RLOCK108	Res.	RLOCK106	RLOCK105	RLOCK104	RLOCK103	RLOCK102	RLOCK101	Res.	Res.	RLOCK98	RLOCK97	RLOCK96
	Reset value															0	0	0	0	0	0		0	0	0	0	0	0			0	0	0
0x030 + 0x8 * x (x=0 to 113) Last addr: 0x3B8	**RCC_RxCIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID [2:0]			Res.	Res.	SEM_EN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034 + 0x8 * x (x=0 to 113) Last addr: 0x3BC	**RCC_RxSEMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x3C0 - 0x3FC	Reserved	Res.																															
0x400	**RCC_GRSTCSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRST
	Reset value																																0
0x404	**RCC_C1RSTCSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1RST
	Reset value																																0
0x408	Reserved	Res.																															
0x40C	**RCC_C2RSTCSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2RST
	Reset value																																0
0x410	**RCC_HWRSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value																		0	0	0		0	0	0	0	0	0	1	0	1	0	1
0x414	**RCC_ C1HWRSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1RSTF	VCPURSTF
	Reset value																															1	1
0x418	**RCC_ C2HWRSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2RSTF
	Reset value																																1
0x41C	**RCC_ C1BOOTRSTSSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	D1STBYRSTF	Res.	STBYC1RSTF	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	Res.	Res.	C1RSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCPURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value									0	0		0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x420	**RCC_ C1BOOTRSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	D1STBYRSTF	Res.	STBYC1RSTF	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	Res.	Res.	C1RSTF	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	VCPURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value									0	0		0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0	0
0x424	**RCC_ C2BOOTRSTSSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	Res.	STBYC2RSTF	Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	Res.	C2RSTF	Res.	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value									0		0		0	0	0			0			0	0	0	0	0	0		0	0	0	0	0
0x428	**RCC_ C2BOOTRSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2STBYRSTF	Res.	STBYC2RSTF	Res.	RETECCFAILRESTRSTF	RETECCFAILCRCRSTF	RETCRCERRRSTF	Res.	Res.	C2RSTF	Res.	Res.	IWDG4SYSRSTF	IWDG3SYSRSTF	IWDG2SYSRSTF	IWDG1SYSRSTF	SYSC2RSTF	SYSC1RSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value									0		0		0	0	0			0			0	0	0	0	0	0		0	0	0	0	0
0x42C	**RCC_C1SREQSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESLPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P0
	Reset value																0																0
0x430	**RCC_C1SREQCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESLPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P0
	Reset value																0																0
0x434	**RCC_CPUBOOTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_CPU1	BOOT_CPU2
	Reset value																															0	0
0x438	**RCC_STBYBOOTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET_CRCERR_RSTEN	Res.	Res.	CPU1_HW_BEN	CPU2_HW_BEN	Res.	COLD_CPU2	CPU_BEN_SEL	Res.
	Reset value																								0			0	0		0	0	

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43C	**RCC_LEGBOOTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEGACY_BEN
	Reset value																																0
0x440	**RCC_BDCR**	VSWRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCCKEN	Res.	Res.	RTCSRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSD	LSEGFON	LSECSSON	LSEDRV[1:0]		LSEDIGBYP	LSERDY	LSEBYP	LSEON
	Reset value	0											0			0	0								0	0	0	1	0	0	0	0	0
0x444 - 0x448	Reserved	Res.																															
0x44C	**RCC_RDCR**	Res.	Res.	Res.	Res.	EADLY[3:0]				Res.	Res.	Res.	MRD[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	1	1	0				0	0	0	0	0																
0x450	**RCC_C1MSRDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1MSRST	Res.	Res.	Res.	C1MSRD[4:0]				
	Reset value																								0				0	0	0	0	0
0x454	**RCC_PWRLPDLYCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPU2TMPSKP	Res.	Res.	PWRLP_DLY[21:0]																					
	Reset value								0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x458	**RCC_C1CIESETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE	Res.	Res.	Res.	PLL8RDYIE	PLL7RDYIE	PLL6RDYIE	PLL5RDYIE	PLL4RDYIE	Res.	PLL2RDYIE	PLL1RDYIE	MSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value												0				0				0	0	0	0	0		0	0	0	0	0	0	0
0x45C	**RCC_C1CIFCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF	Res.	Res.	Res.	PLL8RDYF	PLL7RDYF	PLL6RDYF	PLL5RDYF	PLL4RDYF	Res.	PLL2RDYF	PLL1RDYF	MSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value												0				0				0	0	0	0	0		0	0	0	0	1	0	0
0x460	**RCC_C2CIESETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE	Res.	Res.	Res.	PLL8RDYIE	PLL7RDYIE	PLL6RDYIE	PLL5RDYIE	PLL4RDYIE	Res.	PLL2RDYIE	PLL1RDYIE	MSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value												0				0				0	0	0	0	0		0	0	0	0	0	0	0
0x464	**RCC_C2CIFCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF	Res.	Res.	Res.	PLL8RDYF	PLL7RDYF	PLL6RDYF	PLL5RDYF	PLL4RDYF	Res.	PLL2RDYF	PLL1RDYF	MSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value												0				0				0	0	0	0	0		0	0	0	0	1	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x468	**RCC_ IWDGC1FZSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
	Reset value																															0	0
0x46C	**RCC_ IWDGC1FZCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
	Reset value																															0	0
0x470	**RCC_ IWDGC1CFGSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_KERRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_SYSRSTEN	Res.	IWDG1_SYSRSTEN
	Reset value														0																0		0
0x474	**RCC_ IWDGC1CFGCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_KERRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2_SYSRSTEN	Res.	IWDG1_SYSRSTEN
	Reset value														0																0		0
0x478	**RCC_ IWDGC2FZSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG4	FZ_IWDG3
	Reset value																															0	0
0x47C	**RCC_ IWDGC2FZCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG4	FZ_IWDG3
	Reset value																															0	0
0x480	**RCC_ IWDGC2CFGSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_KERRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_SYSRSTEN	Res.	IWDG3_SYSRSTEN
	Reset value														0																0		0
0x484	**RCC_ IWDGC2CFGCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_KERRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG4_SYSRSTEN	Res.	IWDG3_SYSRSTEN
	Reset value														0																0		0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x488	**RCC_MCO1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO1ON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO1SEL
	Reset value																								0								0
0x48C	**RCC_MCO2CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO2ON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO2SEL
	Reset value																								0								0
0x490	**RCC_OCENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSEKERON	HSEON	HSEDIGBYP	HSEDIV2BYP	HSEDIV2ON	Res.	MSIKERON	MSION	HSIKERON	HSION
	Reset value																					0	0	0	0	0	0	0		0	0	0	1
0x494	**RCC_OCENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEBYP	HSEKERON	HSEON	HSEDIGBYP	HSEDIV2BYP	HSEDIV2ON	Res.	MSIKERON	MSION	HSIKERON	HSION
	Reset value																						0	0	0	0	0	0		0	0	0	1
0x498	**RCC_OCRDYR**	Res.	Res.	Res.	Res.	Res.	Res.	CKREST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERD	Res.	Res.	Res.	Res.	Res.	MSIRDY	Res.	HSIRDY
	Reset value							0																	0						0		1
0x49C	**RCC_HSICFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSICAL[8:0]									Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								
0x4A0	**RCC_MSICFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSICAL[7:0]								Res.	Res.	Res.	MSITRIM[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	1	0	0	0	0				1	0	0	0	0								
0x4A4	**RCC_LSICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value																															0	0
0x4A8	**RCC_RTCDIVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDIV[5:0]					
	Reset value																											0	0	0	0	0	0
0x4AC + 0x4 * (x-1) (x=1 to 5) Last addr: 0x4BC	**RCC_APBxDIVR**	APB1DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBxDIV[2:0]		
	Reset value	1																													0	0	0
0x4C0	**RCC_APBDBGDIVR**	APBDBGDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBDBGDIV[2:0]		
	Reset value	1																													0	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4C4	Reserved	Res.																															
0x4C8	**RCC_TIMG1PRER**	TIMG1PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG1PRE
	Reset value	1																															0
0x4CC	**RCC_TIMG2PRER**	TIMG2PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG2PRE
	Reset value	1																															0
0x4D0	**RCC_LSMCUDIVR**	LSMCUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSMCUDIV
	Reset value	1																															0
0x4D4	**RCC_DDRCPCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRCPLPEN	DDRCPEN	DDRCPRST
	Reset value																													1	0	1
0x4D8	**RCC_ DDRCAPBCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRCAPBLPEN	DDRCAPBEN	DDRCAPBRST
	Reset value																													1	0	1
0x4DC	**RCC_DDRPHYCAPB CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPHYCAPBLPEN	DDRPHYCAPBEN	DDRPHYCAPBRST
	Reset value																													1	0	1
0x4E0	**RCC_ DDRPHYCCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPHYCEN	Res.
	Reset value																														0		

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4E4	**RCC_DDRCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRCFGLPEN	DDRCFGEN	DDRCFGRST
	Reset value																														1	0	1
0x4E8	**RCC_DDRITFCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPHYDLP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRSHR	Res.	Res.	DDRCKMOD[1:0]		Res.	Res.	Res.	DDRRST
	Reset value																0								0			0	0				1
0x4EC	Reserved	Res.																															
0x4F0	**RCC_SYSRAMCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN	SYSRAMEN	Res.
	Reset value																														1	0	
0x4F4	Reserved	Res.																															
0x4F8	**RCC_SRAM1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1LPEN	SRAM1EN	Res.
	Reset value																														1	0	
0x4FC	Reserved	Res.																															
0x500	**RCC_RETRAMCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	RETRAMEN	Res.
	Reset value																														1	1	
0x504	**RCC_ BKPSRAMCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	BKPSRAMEN	Res.
	Reset value																														1	0	
0x508 - 0x510	Reserved	Res.																															
0x514	**RCC_OSPI1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI1DLLRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC1RST	Res.	Res.	Res.	Res.	Res.	OSPI1LPEN	OSPI1EN	OSPI1RST
	Reset value																0								0						1	0	0
0x518	Reserved	Res.																															

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x51C	**RCC_FMCCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCLPEN	FMCEN	FMCRST
	Reset value																														1	0	0
0x520	**RCC_DBGCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGRST	Res.	DBGMCUEN	TRACEEN	DBGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0		0	0	0								
0x524	**RCC_STMCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STMLPEN	STMEN	Res.
	Reset value																														1	0	
0x528	**RCC_ETRCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRLPEN	ETREN	Res.
	Reset value																														1	0	
0x52C + 0x4 * x (x=0 to 8) Last addr: 0x54C	**RCC_GPIOyCFGR (y = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOyLPEN	GPIOyEN	GPIOyRST
	Reset value																														1	0	0
0x550 - 0x554	Reserved	Res.																															
0x558	**RCC_GPIOZCFGR**																														GPIOZLPEN	GPIOZEN	GPIOZRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x55C	**RCC_HPDMA1CFGR**																														HPDMA1LPEN	HPDMA1EN	HPDMA1RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x560	**RCC_HPDMA2CFGR**																														HPDMA2LPEN	HPDMA2EN	HPDMA2RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x564	**RCC_HPDMA3CFGR**																														HPDMA3LPEN	HPDMA3EN	HPDMA3RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x56C	Reserved	Res.																															
0x570	**RCC_IPCC1CFGR**																														IPCC1LPEN	IPCC1EN	IPCC1RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x574	Reserved	Res.																															
0x578	**RCC_RTCCFGR**																														RTCLPEN	RTCEN	Res.
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	
0x57C	Reserved	Res.																															
0x580	**RCC_ SYSCPU1CFGR**																														SYSCPU1LPEN	SYSCPU1EN	Res.
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	
0x584	**RCC_BSECCFGR**																														BSECLPEN	BSECEN	Res.
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	
0x588 - 0x58C	Reserved	Res.																															
0x590	**RCC_PLL2CFGR1**																																
		Res.	Res.	Res.	CKREFST	Res.	Res.	Res.	PLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSMODRST
	Reset value				0				0																0							1
0x594	**RCC_PLL2CFGR2**																																
		Res.	Res.	Res.	Res.	FBDIV[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]				
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x598	**RCC_PLL2CFGR3**																																
		Res.	Res.	Res.	Res.	Res.	SSCGDIS	DACEN	DOWNSPREAD	FRACIN[23:0]																							
	Reset value						1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x59C	**RCC_PLL2CFGR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS	FOUTPOSTDIVEN	DSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																						0	0	0								
0x5A0	**RCC_PLL2CFGR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPREAD[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVVAL [3:0]			
	Reset value												0	0	0	0	0													0	0	0	0
0x5A4	Reserved	Res.																															
0x5A8	**RCC_PLL2CFGR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV1[2:0]		
	Reset value																														0	0	0
0x5AC	**RCC_PLL2CFGR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2[2:0]		
	Reset value																														0	0	0
0x5B0 - 0x5DC	Reserved	Res.																															
0x5E0	**RCC_HSIFMONCR**	HSIMONF	HSIMONIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIDEV[5:0]						HSIMONEN	Res.	Res.	Res.	Res.	HSIREF[10:0]										
	Reset value	0	0									0	1	1	1	1	1	0					1	1	1	1	0	1	0	0	0	0	1
0x5E4	**RCC_HSIFVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIVAL[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5E8	**RCC_MSIFMONCR**	MSIMONF	MSIMONIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSIDEV[4:0]					MSIMONEN	Res.	Res.	Res.	Res.	Res.	Res.	MSIREF[8:0]								
	Reset value	0	0										0	1	1	1	1	0							1	1	1	1	0	1	0	0	0
0x5EC	**RCC_MSIFVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSIVAL[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x5F0 - 0x6FC	Reserved	Res.																															
0x700 + 0x4 * (x-1) (x=1 to 8, 10 to 17) Last addr: 0x73C	**RCC_TIMxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMxLPEN	TIMxEN	TIMxRST
	Reset value																														1	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x740	Reserved	Res.																															
0x744 + 0x4 * (x-1) (x=1 to 5) Last addr: 0x754	**RCC_LPTIMxCFGR**																														LPTIMxLPEN	LPTIMxEN	LPTIMxRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x758 + 0x4 * (x-1) (x=1 to 6) Last addr: 0x76C	**RCC_SPIxCFGR**																														SPIxLPEN	SPIxEN	SPIxRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x770 - 0x774	Reserved	Res.																															
0x778	**RCC_SPDIFRXCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPDIFRXLPEN	SPDIFRXEN	SPDIFRXRST
	Reset value																														1	0	0
0x77C + 0x4 * (x-1) (x=1 to 3) Last addr: 0x784	**RCC_USARTxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USARTxLPEN	USARTxEN	USARTxRST
	Reset value																														1	0	0
0x788 + 0x4 * (x-4) (x=4 to 5) Last addr: 0x78C	**RCC_UARTxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UARTxLPEN	UARTxEN	UARTxRST
	Reset value																														1	0	0
0x790	**RCC_USART6CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART6LPEN	USART6EN	USART6RST
	Reset value																														1	0	0
0x794	**RCC_UART7CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART7LPEN	UART7EN	UART7RST
	Reset value																														1	0	0
0x798 - 0x79C	Reserved	Res.																															

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7A0	**RCC_ LPUART1CFGR**																														LPUART1LPEN	LPUART1EN	LPUART1RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x7A4 + 0x4 * (x-1) (x=1 to 3) Last addr: 0x7AC	**RCC_I2CxCFGR**																														I2CxLPEN	I2CxEN	I2CxRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x7B0 - 0x7C0	Reserved	Res.																															
0x7C4 + 0x4 * (x-1) (x=1 to 4) Last addr: 0x7D0	**RCC_SAIxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAIxLPEN	SAIxEN	SAIxRST
	Reset value																														1	0	0
0x7D4	Reserved	Res.																															
0x7D8	**RCC_MDF1CFGR**																														MDF1LPEN	MDF1EN	MDF1RST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x7DC	Reserved	Res.																															
0x7E0	**RCC_FDCANCFGR**																														FDCANLPEN	FDCANEN	FDCANRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x7E4	**RCC_HDPCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		HDPEN	HDPRST
	Reset value																															0	0
0x7E8	**RCC_ADC1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC1KERSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC1LPEN	ADC1EN	ADC1RST
	Reset value																				0										1	0	0
0x7EC	**RCC_ADC2CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC2KERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC2LPEN	ADC2EN	ADC2RST
	Reset value																			0	0										1	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7F0	**RCC_ETH1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1RXEN	ETH1TXLPEN	ETH1TXEN	Res.	ETH1LPEN	ETH1EN	ETH1STPEN	Res.	ETH1MACLPEN	ETH1MACEN	ETH1RST
	Reset value																					1	0	1	0		1	0	0		1	0	0
0x7F4	**RCC_ETH2CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH2RXLPEN	ETH2RXEN	ETH2TXLPEN	ETH2TXEN	Res.	ETH2LPEN	ETH2EN	ETH2STPEN	Res.	ETH2MACLPEN	ETH2MACEN	ETH2RST
	Reset value																					1	0	1	0		1	0	0		1	0	0
0x7F8	Reserved	Res.																															
0x7FC	**RCC_USBHCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	USBHEN	USBHRST
	Reset value																														1	0	0
0x800	**RCC_ USB2PHY1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1CKREFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1LPEN	USB2PHY1EN	USB2PHY1RST
	Reset value																	0													1	0	0
0x804	Reserved	Res.																															
0x808	**RCC_OTGCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTGLPEN	OTGEN	OTGRST
	Reset value																														1	0	0
0x80C	**RCC_ USB2PHY2CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2CKREFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2LPEN	USB2PHY2EN	USB2PHY2RST
	Reset value																	0													1	0	0
0x810 - 0x820	Reserved	Res.																															
0x824	**RCC_STGENCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	Res.	STGENLPEN	STGENEN	Res.
	Reset value																												0		1	0	

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x828 - 0x82C	Reserved	Res.																															
0x830 + 0x4 * (x-1) (x=1 to 3) Last addr: 0x838	**RCC_SDMMCxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMCxDLLRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMCxLPEN	SDMMCxEN	SDMMCxRST
	Reset value																0														1	0	0
0x83C	Reserved	Res.																															
0x840	**RCC_LTDCCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN	LTDCEN	LTDCRST
	Reset value																														1	0	0
0x844 - 0x854	Reserved	Res.																															
0x858	**RCC_CSICFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSILPEN	CSIEN	CSIRST
	Reset value																														1	0	0
0x85C	**RCC_DCMIPPCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPPLPEN	DCMIPPEN	DCMIPPRST
	Reset value																														1	0	0
0x860	**RCC_ DCMIPSSICFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPSSILPEN	DCMIPSSIEN	DCMIPSSIRST
	Reset value																														1	0	0
0x864 - 0x86C	Reserved	Res.																															
0x870 + 0x4 * (x-1) (x=1 to 2) Last addr: 0x874	**RCC_RNGxCFGR**																														RNGxLPEN	RNGxEN	RNGxRST
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	0
0x878	**RCC_PKACFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKALPEN	PKAEN	PKARST
	Reset value																														1	0	0
0x87C	**RCC_SAESCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESLPEN	SAESEN	SAESRST
	Reset value																														1	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x880 + 0x4 * (x-1) (x=1 to 2) Last addr: 0x884	**RCC_HASHxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHxLPEN	HASHxEN	HASHxRST
	Reset value																														1	0	0
0x888 + 0x4 * (x-1) (x=1 to 2) Last addr: 0x88C	**RCC_CRYPxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPxLPEN	CRYPxEN	CRYPxRST
	Reset value																														1	0	0
0x890	Reserved	Res.																															
0x894 + 0x4 * (x-1) (x=1 to 4) Last addr: 0x8A0	**RCC_IWDGxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDGxLPEN	IWDGxEN	Res.
	Reset value																														1	0	
0x8A4	**RCC_WWDG1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1LPEN	WWDG1EN	WWDG1RST
	Reset value																														1	0	0
0x8A8	Reserved	Res.																															
0x8AC	**RCC_VREFCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFLPEN	VREFEN	VREFRST
	Reset value																														1	0	0
0x8B0	**RCC_DTSCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSKERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	DTSEN	DTSRST
	Reset value																			0	0										1	0	0
0x8B4	**RCC_CRCCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	CRCEN	CRCRST
	Reset value																														1	0	0
0x8B8	**RCC_SERCCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SERCLPEN	SERCEN	SERCRST
	Reset value																														1	0	0
0x8BC	Reserved	Res.																															

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8C0	**RCC_ DDRPERFMCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	DDRPERFMEN	DDRPERFMRST
	Reset value																														1	0	0
0x8C4	Reserved	Res.																															
0x8C8 + 0x4 * (x-1) (x=1 to 3) Last addr: 0x8D0	**RCC_I3CxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3CxLPEN	I3CxEN	I3CxRST
	Reset value																														1	0	0
0x8D4 - 0xFFC	Reserved	Res.																															
0x1000	**RCC_MUXSELCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	MUXSEL6[1:0]		Res.	Res.	MUXSEL5[1:0]		Res.	MUXSEL4[2:0]			Res.	MUXSEL3[2:0]			Res.	MUXSEL2[2:0]			Res.	MUXSEL1[2:0]			Res.	MUXSEL0[2:0]		
	Reset value							0	0			0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x1004 - 0x1014	Reserved																																
0x1018 + 0x4 *x (x=1 to 63) Last add: 0x1114	**RCC_XBARxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XBARxSTS	XBARxEN	Res.	Res.	XBARxSEL[3:0]			
	Reset value																									0	1			0	1	0	1
0x1118 + 0x4 *x (x=1 to 63) Last add: 0x1214	**RCC_PREDIVxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIVx[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1218	**RCC_PREDIVSR1**	PREDIVSTS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x121C	**RCC_PREDIVSR2**	PREDIVSTS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1220	Reserved	Res.																															
0x1224 + 0x4 *x (x=1 to 63) Last add: 0x1320	**RCC_FINDIVxCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINDIVxEN	FINDIVx[5:0]					
	Reset value																										1	0	0	0	0	0	0
0x1324	**RCC_FINDIVSR1**	FINDIVSTS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1328	**RCC_FINDIVSR2**	FINDIVSTS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x132C - 0x133C	Reserved	Res.																															
0x1340	**RCC_ FCALCOBS0CFGR**	Res.	Res.	Res.	Res.	Res.	CKOBSEN	FCALCCKEN	CKOBSDIV[2:0]			Res.	Res.	Res.	CKOBSINV	FCALCCKINV	CKOBSEXTSEL	FCALCCKEXTSEL	Res.	Res.	Res.	Res.	CKEXTSEL[2:0]			CKINTSEL[7:0]							
	Reset value						1	1	0	0	0				0	0	0	0					0	0	0	1	0	0	0	0	1	1	1
0x1344	**RCC_ FCALCOBS1CFGR**	Res.	Res.	Res.	Res.	FCALCRSTN	CKOBSEN	Res.	CKOBSDIV[2:0]			Res.	Res.	Res.	CKOBSINV	Res.	CKOBSEXTSEL	Res.	Res.	Res.	Res.	Res.	CKEXTSEL[2:0]			CKINTSEL[7:0]							
	Reset value					0	1		0	0	0				0		0						0	0	0	1	0	0	0	0	1	1	1
0x1348	**RCC_ FCALCREFCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCREFCKSEL[2:0]		
	Reset value																														0	0	0
0x134C	**RCC_FCALCCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCRUN
	Reset value																																0
0x1350	Reserved	Res.																															
0x1354	**RCC_FCALCCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCTYP[4:0]					Res.	Res.	FCALCTWC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCMD[1:0]		Res.	Res.	Res.
	Reset value											0	0	0	0	0			0	0	0	0							0	0			
0x1358	**RCC_FCALCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCALCSTS	Res.	Res.	FVAL[16:0]																
	Reset value													0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x135C	Reserved	Res.																															
0x1360 +0x28 * (y-4) (y=4 to 8) Last addr: 0x1400	**RCC_PLLyCFGR1**	Res.	Res.	Res.	CKREFST	Res.	Res.	Res.	PLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSMODRST
	Reset value				1				0																0								1

Table 162. RCC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1364 +0x28 * (y-4) (y=4 to 8) Last addr: 0x1404	**RCC_PLLyCFGR2**	Res.	Res.	Res.	Res.	FBDIV[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREFDIV[5:0]					
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x1368 +0x28 * (y-4) (y=4 to 8) Last addr: 0x1408	**RCC_PLLyCFGR3**	Res.	Res.	Res.	Res.	Res.	SSCGDIS	DACEN	DOWNSPREAD	FRACIN[23:0]																							
	Reset value						1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x136C +0x28 * (y-4) (y=4 to 8) Last addr: 0x140C	**RCC_PLLyCFGR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS	FOUTPOSTDIVEN	DSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																						0	0	0								
0x1370 +0x28 * (y-4) (y=4 to 8) Last addr: 0x1410	**RCC_PLLyCFGR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPREAD[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVVAL [3:0]			
	Reset value												0	0	0	0	0													0	0	0	0
0x1374	Reserved	Res.																															
0x1378 +0x28 * (y-4) (y=4 to 8) Last addr: 0x1418	**RCC_PLLyCFGR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV1[2:0]		
	Reset value																														0	0	0
0x137C +0x28 * (y-4) (y=4 to 8) Last addr: 0x141C	**RCC_PLLyCFGR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSTDIV2[2:0]		
	Reset value																														0	0	0
0x1420 - 0xFFF0	Reserved	Res.																															
0xFFF4	**RCC_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0xFFF8	**RCC_IDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0xFFFC	**RCC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0

21 Inter-processor communication controller (IPCC)

21.1 Introduction

The inter-processor communication controller (IPCC) is used for communicating data between two processors.

The IPCC block provides a nonblocking signaling mechanism to post and retrieve communication data in an atomic way. It provides the signaling for 32 channels:

- 16 channels in the direction from processor 1 to processor 2
- 16 channels in the opposite direction

It is then possible to have two different communication types in each direction.

The IPCC communication data must be located in a common memory, which is not part of the IPCC block.

21.2 IPCC main features

- Status signaling for the 32 channels
  - Channel occupied/free flag, also used as lock
- Four interrupt lines per processor
  - One nonsecure and one secure for RX channel occupied (communication data posted by sending processor)
  - One nonsecure and one secure for TX channel free (communication data retrieved by receiving processor)
- Interrupt masking per channel
  - Channel occupied mask
  - Channel free mask
- Two channel operation modes
  - Simplex (each channel has its own communication data memory location)
  - Half duplex (a single channel associated with a bidirectional communication data information memory location)
- Security support
  - Support for AHB independent secure and nonsecure IPCC channels. Protecting any secure resource (register field) from a nonsecure access
  - Generation of separate secure and nonsecure interrupt per CPU
- Privileged/unprivileged support
  - Support for AHB independent privilege and unprivileged IPCC channels. Protecting any privileged resource (register field) from an unprivileged access
- Resource isolation support
- CID-aware register access. Protecting any CID domain (register) from access by another CID domain.
  - Dedicated processor interrupt lines per CID domain and interrupt routing.

21.3 IPCC functional description

The IPCC communication data are located in a common memory, which is not part of the IPCC block. The address location of the communication data must be known or located in a known common area that, as already stated, is not part of the IPCC block.

For each communication, the IPCC block provides a channel status flag CHnF.

- When 0, the channel status flag CHnF indicates that the associated IPCC channel is free (the receiving processor has retrieved communication data), and can be accessed by the sending processor.
- When 1, the channel status flag CHnF indicates that the associated IPCC channel is occupied (the sending processor has posted communication data) and can be accessed by the receiving processor.

The channel operation mode must be known to both processors. A common parameter can be used to indicate the channel transfer mode and must also be located in a known common area. This parameter is not available from the IPCC.

21.3.1 IPCC block diagram

The IPCC (see *Figure 122*) consists of the following subblocks:

- Status block, containing the channel status
- IPCC interface block, providing AHB access to the channel status registers
- Interrupt interface block, providing control for the interrupts

Figure 122. IPCC block diagram

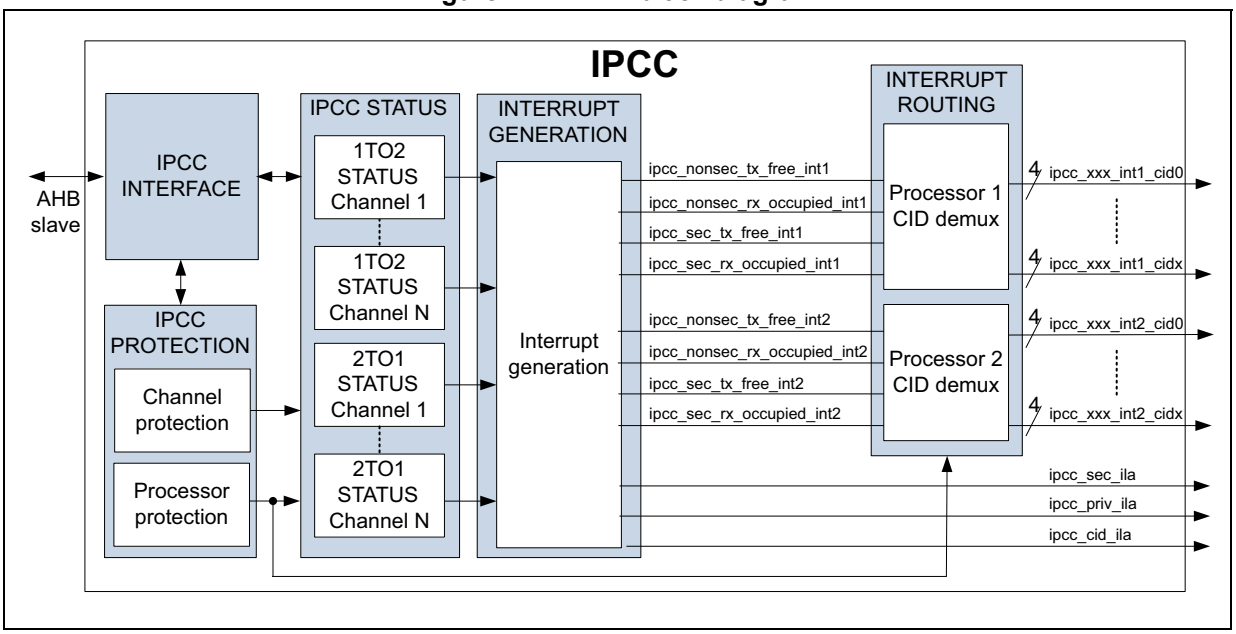


Table 163. IPCC interface signals

Signal		Description
Name	Type	
AHB slave	I/O	AHB register access bus
ipcc_nonsec_tx_free_int1_cidx	O	Nonsecure TX free interrupt to processor 1 per CID
ipcc_nonsec_rx_occupied_int1_cidx	O	Nonsecure RX occupied interrupt to processor 1 per CID
ipcc_sec_tx_free_int1_cidx	O	Secure TX free interrupt to processor 1 per CID
ipcc_sec_rx_occupied_int1_cidx	O	Secure RX occupied interrupt to processor 1 per CID
ipcc_nonsec_tx_free_int2_cidx	O	Nonsecure TX free interrupt to processor 2 per CID
ipcc_nonsec_rx_occupied_int2_cidx	O	Nonsecure RX occupied interrupt to processor 2 per CID
ipcc_sec_tx_free_int2_cidx	O	Secure TX free interrupt to processor 2 per CID
ipcc_sec_rx_occupied_int2_cidx	O	Secure RX occupied interrupt to processor 2 per CID
ipcc_sec_ila	O	Illegal secure access event
ipcc_priv_ila	O	Illegal privilege access event
ipcc_cid_ila	O	Illegal CID access event

21.3.2 IPCC Simplex channel mode

In Simplex channel mode, a dedicated memory location (used to transfer data in a single direction) is assigned to the communication data. The associated channel N control bits (see *Table 164*) are used to manage the transfer from the sending to the receiving processor.

Table 164. Bits used for the communication

Processor	A	B
SEND A = 1 RECEIVE B = 2	IPCC_C1CR.TXFIE IPCC_C1MR.CHnFM IPCC_C1SCR.CHnS IPCC_C1TOC2SR.CHnF	IPCC_C2CR.RXOIE IPCC_C2MR.CHnOM IPCC_C2SCR.CHnC
SEND A = 2 RECEIVE B = 1	IPCC_C2CR.TXFIE IPCC_C2MR.CHnFM IPCC_C2SCR.CHnS IPCC_C2TOC1SR.CHnF	IPCC_C1CR.RXOIE IPCC_C1MR.CHnOM IPCC_C1SCR.CHnC

Once the sending processor has posted the communication data in the memory, it sets the channel status flag CHnF to occupied with CHnS.

Once the receiving processor has retrieved the communication data from the memory, it clears the channel status flag CHnF back to free with CHnC.

Figure 123. IPCC Simplex channel mode transfer timing

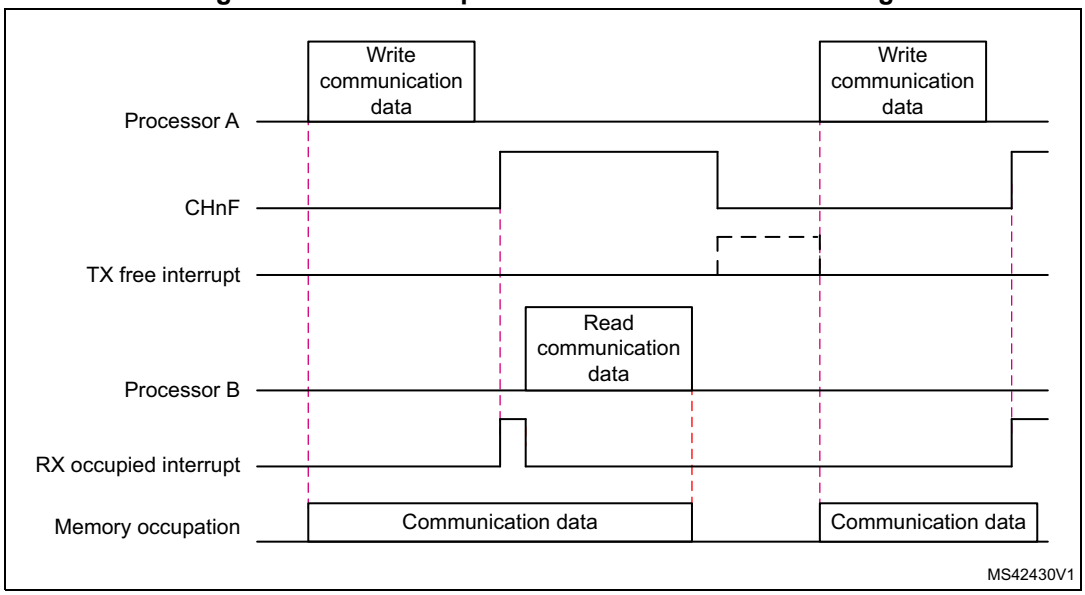
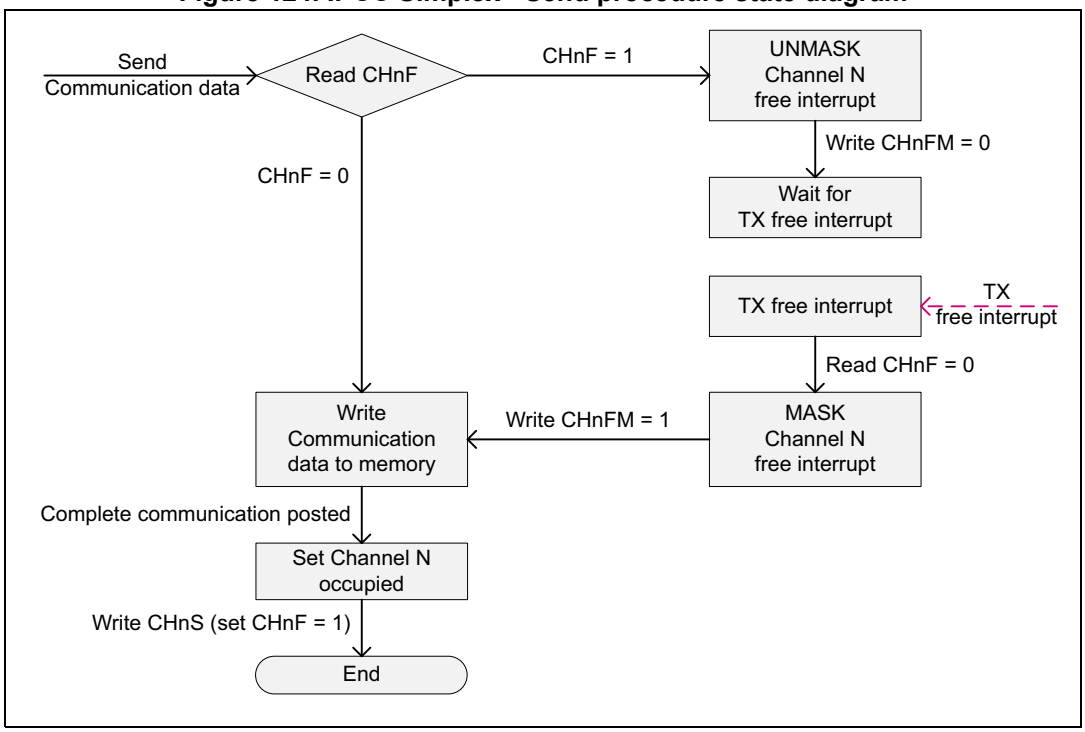


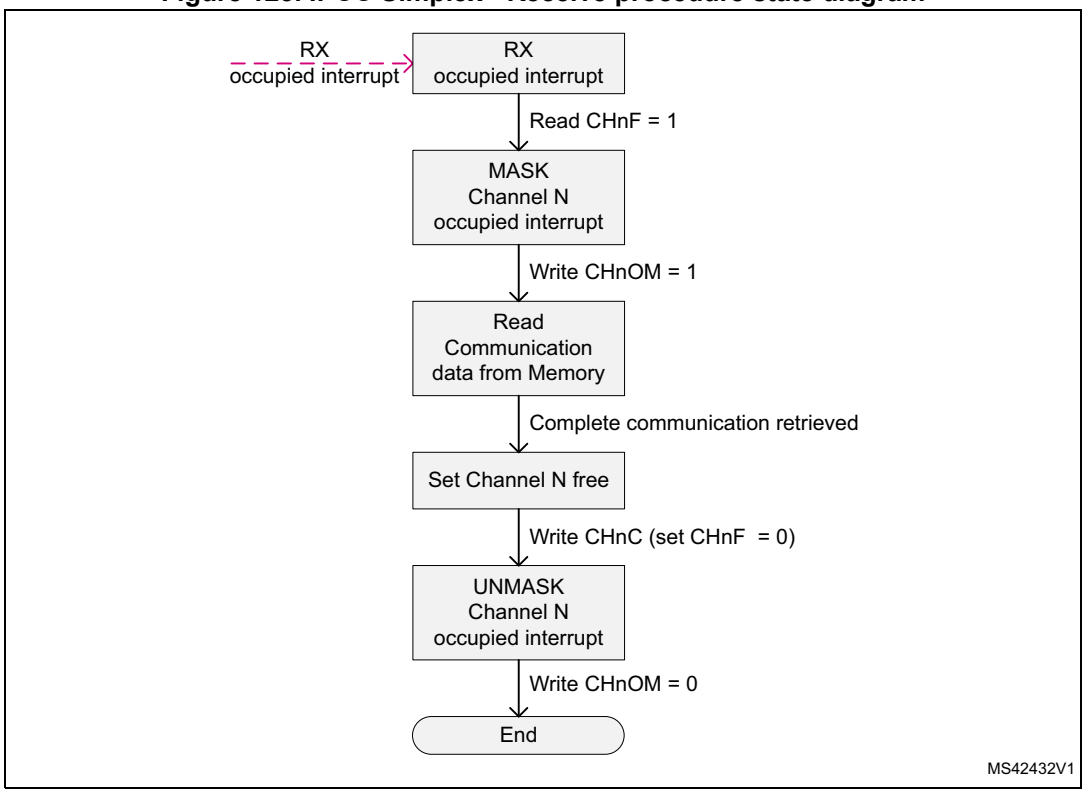
Figure 124. IPCC Simplex - Send procedure state diagram



To send communication data:

- The sending processor checks the channel status flag CHnF:
  - When CHnF = 0, the channel is free (the last communication data is retrieved by the receiving processor), and the new communication data can be written.
  - When CHnF = 1, the channel is occupied (the last communication data is not retrieved by the receiving processor), and the sending processor unmasks the channel free interrupt (CHnFM = 0).
  - On a TX free interrupt, the sending processor checks which channel became free and masks the channel free interrupt (CHnFM = 1). Then the new communication can take place.
- Once the complete communication data is posted, the channel status is set to occupied with CHnS. This gives memory access to the receiving processor and generates the RX occupied interrupt.

Figure 125. IPCC Simplex - Receive procedure state diagram



To receive a communication, the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from memory.
- Once the complete communication data is retrieved, the channel status is cleared to free with CHnC. This gives memory access back to the sending processor and may generate the TX free interrupt.
- Once the channel status is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

21.3.3 IPCC Half-duplex channel mode

The Half-duplex channel mode is used when one processor sends a communication and the other processor sends a response to each communication (ping-pong).

In Half-duplex channel mode, a single dedicated memory location is assigned to communication data and response, and is used to transfer data in both directions. The sending processor channel status flag CHnF is assigned to the channel and used by both processors (see *Table 164*).

Once the processor A posts communication data into memory, it sets the processor A channel status flag CHnF to occupied with CHnS (giving memory access to processor B).

Once the processor B retrieves communication data from memory, it does not change the channel status flags. The memory access is kept by processor B for the response.

Once the processor B posts the response into memory, it clears the channel status flag CHnF to free with CHnC (giving memory access back to processor A).

Once the processor A retrieves the response from the memory, it does not change the channel status flags. The memory location access is kept by processor A for the next communication data.

Figure 126. IPCC Half-duplex channel mode transfer timing

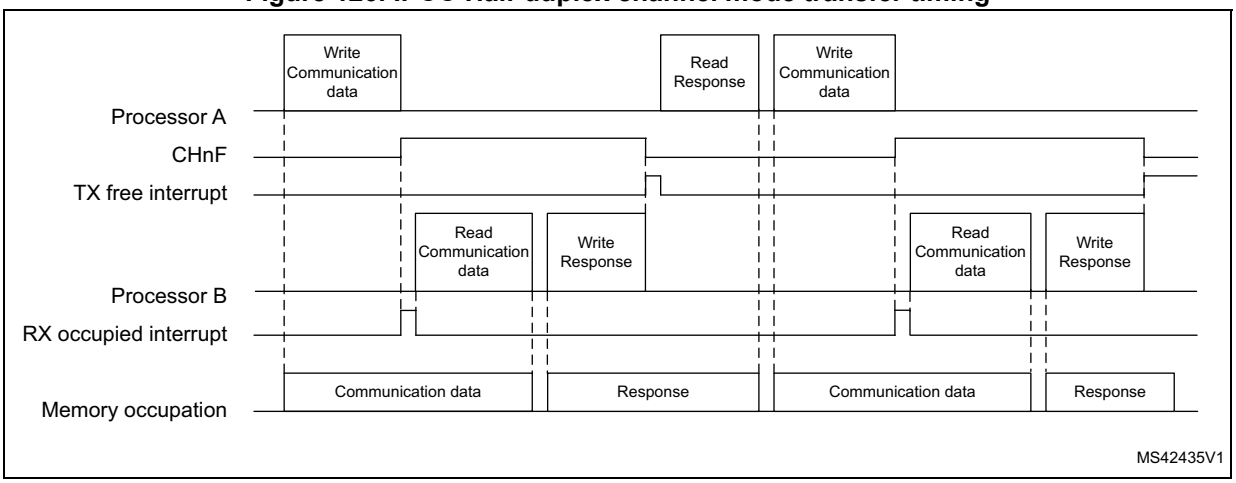
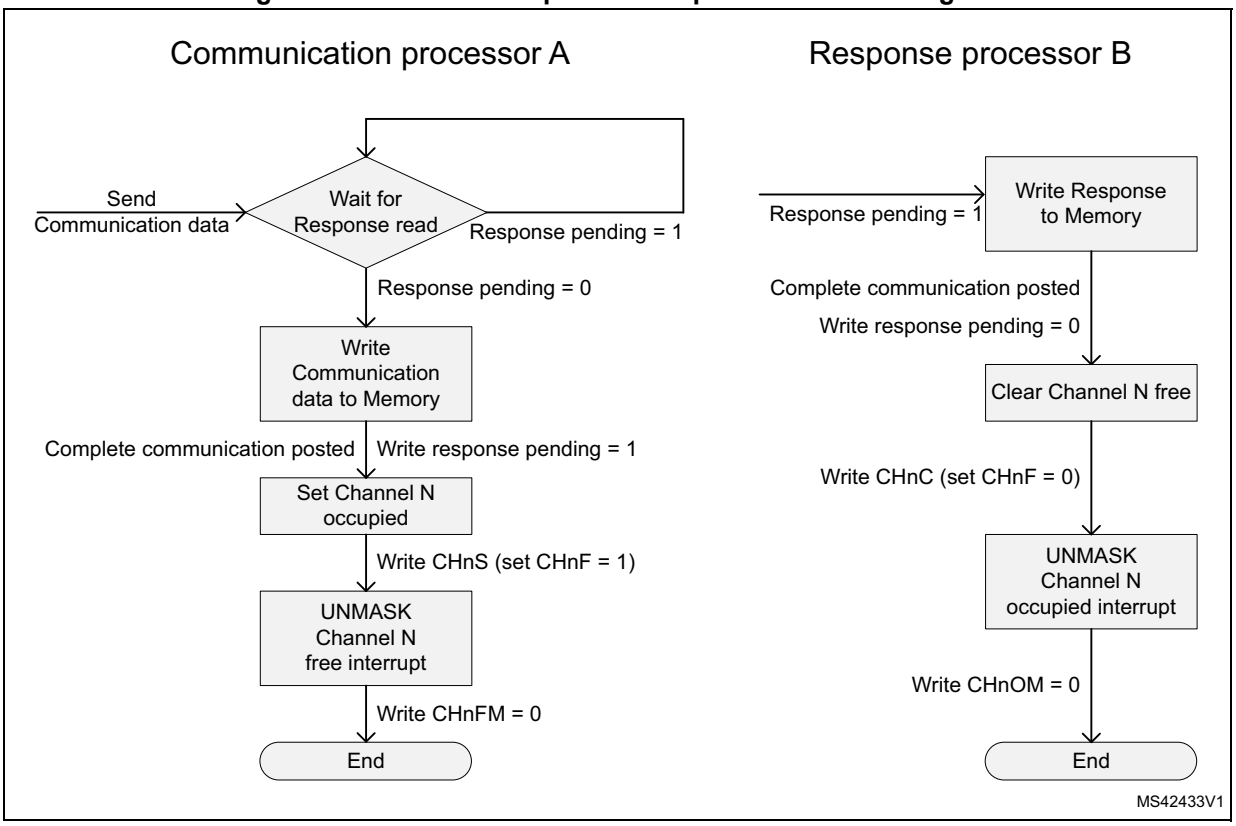


Figure 127. IPCC Half-duplex - Send procedure state diagram



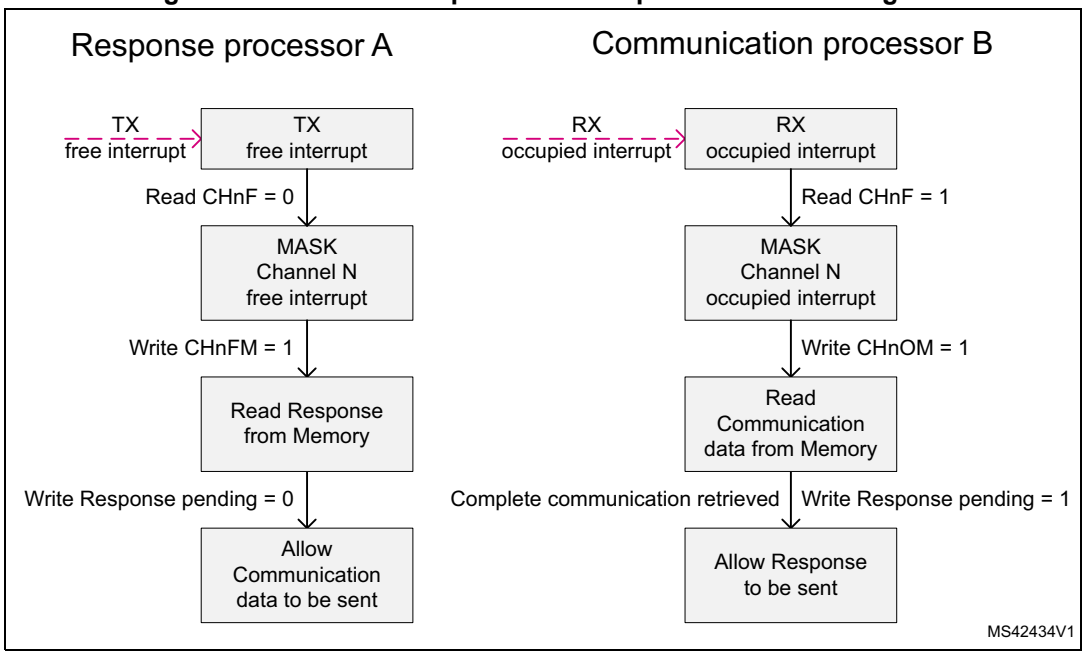
To send communication data:

- The sending processor waits for its response pending software variable to get 0.
  - Once the response pending software variable is 0 the communication data is posted.
- Once the complete communication data has been posted, the channel status flag CHnF is set to occupied with CHnS, and the response pending software variable is set to 1 (this gives memory access and generates the RX occupied interrupt to the receiving processor).
- Once the channel status flag CHnF is set, the channel free interrupt is unmasked (CHnFM = 0).

To send a response:

- The receiving processor waits for its response pending software variable to get 1.
  - Once the response pending software variable is 1, the response is posted.
- Once the complete response is posted, the channel status flag CHnF is cleared to free with CHnC, and the response pending software variable is set to 0 (this gives memory access and generates the TX free interrupt to the sending processor).
- Once the channel status flag CHnF is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

Figure 128. IPCC Half-duplex - Receive procedure state diagram



To receive communication data the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from the memory.
- Once the complete communication data is retrieved, the response pending software variable is set. The channel status is not changed, access to the memory is kept to post the subsequent response.

To receive the response the channel free interrupt is unmasked (CHnFM = 0):

- On a TX free interrupt, the sending processor checks which channel became free, masks the associated channel free interrupt (CHnFM) and reads the response from the memory.
- Once the complete response is retrieved, the response pending software variable is cleared. The channel status is not changed, access to the memory is kept to post the subsequent communication data.

21.3.4 IPCC interrupts and events

There are eight interrupt lines per CID and security events:

- Four RX channel occupied interrupts, one nonsecure and one secure for each processor: CID compartment
  - Interrupt enable RXOIE, SECRXOIE per processor
  - Individual mask CHnOM per channel
  - Interrupt routing based on processor CID in IPCC_CxCIDCFGR
- Four TX channel free interrupts, one nonsecure and one secure for each processor CID compartment
  - Interrupt enable TXFIE, SECTXFIE per processor
  - Individual mask CHnFM per channel
  - Interrupt routing based on processor CID in IPCC_CxCIDCFGR
- Illegal access events, for security, privilege, and CID

The RX occupied interrupt is used by the receiving processor and indicates when an unmasked channel status indicates occupied (CHnF = 1).

The TX free interrupt is used by the sending processor, and indicates when an unmasked channel status indicates free (CHnF = 0).

A secure channel only generates a secure interrupt, and only in the case when the channel is secure unmasked and global secure enabled.

A nonsecure channel only generates a nonsecure interrupt, and only in the case when the channel is nonsecure unmasked and global nonsecure enabled.

Interrupt routing

When CID filtering is enabled in IPCC_CxCIDCFGR.CFEN, the associated processor x interrupt is only routed to the enabled CID.

When CID filtering is disabled in IPCC_CxCIDCFGR.CFEN, the associated processor x interrupt is broadcasted to all supported CIDs. To prevent a compartment to receive unwanted IPCC interrupts, it must be masked at the processor compartment level (in the CPU NVIC).

21.3.5 IPCC security

The IPCC provides per processor and per channel security, in the IPCC_C1SECCFGR, IPCC_C1PRIVCFGR, IPCC_C2SECCFGR and IPCC_C2PRIVCFGR registers. This makes possible to configure any channel in one of the modes detailed in *Table 165*.

Table 165. IPCC security modes

Processor 1	Processor 2	Description
Nonsecure	Nonsecure	Nonsecure / unprivileged
Nonsecure	Secure	TX secure - RX nonsecure / unprivileged
Secure	Nonsecure	TX nonsecure - RX secure / unprivileged
Secure	Secure	Full secure / unprivileged
Nonsecure privilege	Nonsecure	Nonsecure TX privilege

Table 165. IPCC security modes (continued)

Processor 1	Processor 2	Description
Nonsecure	Nonsecure privilege	Nonsecure RX privilege
Nonsecure privilege	Nonsecure privilege	Nonsecure full privilege
Nonsecure privilege	Secure	TX secure privilege - RX nonsecure
Nonsecure	Secure privilege	TX secure - RX nonsecure privilege
Nonsecure privilege	Secure privilege	TX secure - RX nonsecure / Full privilege
Secure privilege	Nonsecure	TX nonsecure privilege - RX secure
Secure	Nonsecure privilege	TX nonsecure - RX secure privilege
Secure privilege	Nonsecure privilege	TX nonsecure -RX secure/ Full privilege
Secure privilege	Secure	TX privileged - RX unprivilege / Full secure
Secure	Secure privilege	TX unprivileged - RX privilege / Full secure
Secure privilege	Secure privilege	Full secure privilege

All processor nonsecure channels generate a processor nonsecure interrupt. All processor secure channels generate an associated processor secure interrupt. Illegal accesses to the security control register and privileged control registers are signaled on the illegal access events.

Additionally, the resource isolation allows the processor 1 registers to be protected from the processor 2 accesses and vice versa, and from any other AHB bus master access. Illegal CID processor accesses are signaled on the CID illegal access event.

The channel status in IPCC_CxTOCySR registers are readable by both processors and by all secure and privileged protection levels of these processors. For this the access protection on these registers is only provided on the compartment filtering when enabled for both processor 1 and processor 2. The following table summarizes the protection on IPCC_CxTOCySR. Not having security and privileged protection allows channels to be shared between secure and nonsecure, privileged, and unprivileged compartments.

Table 166. Access to channels

IPCC_C1CIDCFGR	IPCC_C2CIDCFGR	IPCC_CxTOCySR access
CFEN = 0	CFEN = 0	No protection, access by secure, nonsecure, privileged, unprivileged and any CID.
CFEN = 1	CFEN = 0	
CFEN = 0	CFEN = 1	
CFEN = 1	CFEN = 1	Compartment isolation protected, only access by processor 1 and processor 2 CIDs with secure, nonsecure, privileged, and unprivileged. Other CID accesses are illegal.

21.4 IPCC registers

The peripheral registers must be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses are not permitted and do not generate a bus error.

For any DDCID access to registers and register bits, a security and privileged policy applies. When DDEN = 0 accesses from the DDCID to CID filtered registers is ignored. When DDEN = 1 the DDCID has access to all registers regardless of any CID filtering policy.

21.4.1 IPCC processor 1 control register (IPCC_C1CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is CID-filtered depending on the CID configuration in the processor 1 IPCC_C1CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC TXFIE	TXFIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC RXOIE	RXOIE
														rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SECTXFIE**: Processor 1 transmit channel free secure interrupt enable.
Associated with IPCC_C1TOC2SR.
This bit can be accessed only with a secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
1: Enable an unmasked processor 1 transmit channel free to generate a TX free secure interrupt.
0: Processor 1 TX free secure interrupt disabled.

Bit 16 **TXFIE**: Processor 1 transmit channel free nonsecure interrupt enable
Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 1 transmit channel free to generate a nonsecure TX free interrupt.
0: Processor 1 nonsecure TX free interrupt disabled

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **SECRXOIE**: Processor 1 receive channel occupied secure interrupt enable.
Associated with IPCC_C2TOC1SR
This bit can be accessed only with a secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied secure interrupt.
0: Processor 1 RX occupied secure interrupt disabled.

Bit 0 **RXOIE**: Processor 1 receive channel occupied interrupt enable
Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied interrupt.
0: Processor 1 RX occupied interrupt disabled

21.4.2 IPCC processor 1 mask register (IPCC_C1MR)

Address offset: 0x004

Reset value: 0xFFFF FFFF

This register is CID-filtered depending on the CID configuration in the processor 1 IPCC_C1CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH16 FM	CH15 FM	CH14 FM	CH13 FM	CH12 FM	CH11 FM	CH10 FM	CH9 FM	CH8 FM	CH7 FM	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 OM	CH15 OM	CH14 OM	CH13 OM	CH12 OM	CH11 OM	CH10 OM	CH9 OM	CH8 OM	CH7 OM	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CHnFM:** Processor 1 transmit channel n status set, (n = 16 to 1).
Associated with IPCC_C1TOC2SR.CHnF
When processor 1 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 1 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 1 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 1 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:0 **CHnOM**: Processor 1 receive channel n status clear (n = 16 to 1).
Associated with IPCC_C2TOC1SR.CHnF
When processor 1 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 1 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 1 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 1 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

21.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR)

Address offset: 0x008

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

This register is CID-filtered depending on the CID configuration in the processor 1 IPCC_C1CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH16S	CH15S	CH14S	CH13S	CH12S	CH11S	CH10S	CH9S	CH8S	CH7S	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16C	CH15C	CH14C	CH13C	CH12C	CH11C	CH10C	CH9C	CH8C	CH7C	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CHnS:** Processor 1 transmit channel n status set (n = 16 to 1).
Associated with IPCC_C1TOC2SR.CHnF
When processor 1 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 1 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 1 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 1 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Processor 1 transmit channel n status bit set.
0: No action.

Bits 15:0 **CHnC**: Processor 1 receive channel n status clear (n = 16 to 1).
Associated with IPCC_C2TOC1SR.CHnF
When processor 1 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 1 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 1 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 1 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Processor 1 receive channel n status bit clear.
0: No action.

21.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR)

Address offset: 0x00C

Reset value: 0x0000 0000

When processor 1 or processor 2 CID filtering is disabled, this register has no access limitations, and can be accessed by any domain CID, and any combination of secure, nonsecure, privilege, and unprivileged access.

When both processor 1 and processor 2 CID filtering is enabled. Access restriction applies and only both processors CIDs have access, other CID accesses are discarded. In this case, an illegal CID read access returns zero data. A CID illegal access event is generated also on illegal accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16F	CH15F	CH14F	CH13F	CH12F	CH11F	CH10F	CH9F	CH8F	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnF:** Processor 1 transmit to processor 2 receive channel n status flag before masking (n = 16 to 1).

1: Channel occupied, data can be read by the receiving processor 2.
When channel is nonsecure in C2SECCFGR, generates a nonsecure RX occupied interrupt to processor 2, when unmasked and global nonsecure enabled for processor 2.
When channel is secure in C2SECCFGR, generates a secure RX occupied interrupt to processor 2, when unmasked and global secure enabled for processor 2.
0: Channel free, data can be written by the sending processor 1.
When channel is nonsecure in C1SECCFGR, generates a nonsecure TX free interrupt to processor 1, when unmasked and global nonsecure enabled for processor 1.
When channel is secure in C1SECCFGR, generates a secure TX free interrupt to processor 1, when unmasked and global secure enabled for processor 1.

21.4.5 IPCC processor 2 control register (IPCC_C2CR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is CID-filtered depending on the CID configuration in the processor 2 IPCC_C2CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC TXFIE	TXFIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC RXOIE	RXOIE
														rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SECTXFIE:** Processor 2 secure transmit channel free interrupt enable.
Associated with IPCC_C2TOC1SR.
This bit can be accessed only with a secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
1: Enable an unmasked processor 2 transmit channel free to generate a secure TX free interrupt.
0: Processor 2 secure TX free interrupt disabled.

Bit 16 **TXFIE:** Processor 2 transmit channel free interrupt enable
Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 2 transmit channel free to generate a TX free interrupt.
0: Processor 2 TX free interrupt disabled

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **SECRXOIE:** Processor 2 secure Receive channel occupied interrupt enable.
Associated with IPCC_C1TOC2SR.
This bit can be accessed only with a secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
1: Enable an unmasked processor 2 receive channel occupied to generate an secure RX occupied interrupt.
0: Processor 2 secure RX occupied interrupt disabled.

Bit 0 **RXOIE:** Processor 2 receive channel occupied interrupt enable
Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 2 receive channel occupied to generate an RX occupied interrupt.
0: Processor 2 RX occupied interrupt disabled

21.4.6 IPCC processor 2 mask register (IPCC_C2MR)

Address offset: 0x014

Reset value: 0xFFFF FFFF

This register is CID-filtered depending on the CID configuration in the processor 2 IPCC_C2CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH16 FM	CH15 FM	CH14 FM	CH13 FM	CH12 FM	CH11 FM	CH10 FM	CH9 FM	CH8 FM	CH7 FM	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 OM	CH15 OM	CH14 OM	CH13 OM	CH12 OM	CH11 OM	CH10 OM	CH9 OM	CH8 OM	CH7 OM	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CHnFM:** Processor 2 transmit channel n free interrupt mask (n = 16 to 1).
Associated with IPCC_C2TOC1SR.CHnF
When processor 2 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 2 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 2 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 2 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:0 **CHnOM**: Processor 2 receive channel n occupied interrupt mask (n = 16 to 1).
Associated with IPCC_C1TOC2SR.CHnF
When processor 2 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 2 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 2 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 2 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

21.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

This register is CID-filtered depending on the CID configuration in the processor 2 IPCC_C2CIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses a CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH16S	CH15S	CH14S	CH13S	CH12S	CH11S	CH10S	CH9S	CH8S	CH7S	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16C	CH15C	CH14C	CH13C	CH12C	CH11C	CH10C	CH9C	CH8C	CH7C	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CHnS:** Processor 2 transmit channel n status set (n = 16 to 1).
Associated with IPCC_C2TOC1SR.CHnF
When processor 2 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 2 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 2 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 2 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Processor 2 transmit channel n status bit set.
0: No action.

Bits 15:0 **CHnC**: Processor 2 receive channel n status clear (n = 16 to 1).
Associated with IPCC_C1TOC2SR.CHnF
When processor 2 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 2 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded, nonsecure read returns 0 value.
When processor 2 CHnPRIV is disabled, bit n can be accessed with privilege and unprivileged access.
When processor 2 CHnPRIV is enabled, bit n can be accessed only with privilege access. Unprivileged write to this bit is discarded, unprivileged read returns 0 value.
1: Processor 2 receive channel n status bit clear.
0: No action.

21.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR)

Address offset: 0x01C

Reset value: 0x0000 0000

When processor 1 or processor 2 CID filtering is disabled, this register has no access limitations, and can be accessed by any domain CID, and by any combination of secure, nonsecure, privileged, and unprivileged access.

When both processor 1 and processor 2 CID filtering is enabled, access restriction applies and only CIDs of both processors have access, other CID accesses are discarded. In this case, an illegal CID read access returns zero data. A CID illegal access event is generated also on illegal accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16F	CH15F	CH14F	CH13F	CH12F	CH11F	CH10F	CH9F	CH8F	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnF:** Processor 2 transmit to processor 1 receive channel n status flag before masking (n = 16 to 1)

1: Channel occupied, data can be read by the receiving processor 1.
When channel is nonsecure in C1SECCFGR, generates a nonsecure RX occupied interrupt to processor 1, when unmasked and global nonsecure enabled for processor 1.
When channel is secure in C1SECCFGR, generates a secure RX occupied interrupt to processor 1, when unmasked and global secure enabled for processor 1.
0: Channel free, data can be written by the sending processor 2.
When channel is nonsecure in C2SECCFGR, generates a nonsecure TX free interrupt to processor 2, when unmasked and global nonsecure enabled for processor 2.
When channel is secure in C2SECCFGR, generates a secure TX free interrupt to processor 2, when unmasked and global secure enabled for processor 2.

21.4.9 IPCC processor 1 security configuration register (IPCC_C1SECCFGR)

Address offset: 0x080

Reset value: 0x0000 0000

This register is CID-filtered depending on the CID configuration in the processor 1 IPCC_C1CIDCFGR. When CID filtering is enabled this register can be accessed only by the enabled CID. An illegal CID write access is discarded and a read access returns zero data, on illegal accesses a CID illegal access event is generated.

This register can only be write accessed by a secure privileged access. A nonsecure or unprivileged write access is discarded and causes the generation of an illegal access event. All CID filtered read accesses return the register data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 SEC	CH15 SEC	CH14 SEC	CH13 SEC	CH12 SEC	CH11 SEC	CH10 SEC	CH9 SEC	CH8 SEC	CH7 SEC	CH6 SEC	CH5 SEC	CH4 SEC	CH3 SEC	CH2 SEC	CH1 SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnSEC:** Processor 1 channel n security enable (n = 16 to 1).

1: Channel n processor 1 security enabled (secure)
0: Channel n processor 1 security disabled (nonsecure).

21.4.10 IPCC processor 1 privilege configuration register (IPCC_C1PRIVCFGR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is CID-filtered, depending upon the CID configuration in the processor 1 IPCC_C1CIDCFGR. When CID filtering is enabled, this register can be accessed only by the enabled CID. An illegal CID write access is discarded, and a read access returns zero

data. A CID illegal access event is generated on illegal accesses.

This register can only be write accessed by a privilege access. An unprivileged write access is discarded and causes the generation of a privilege illegal access event. All CID filtered read accesses return the register data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 PRIV	CH15 PRIV	CH14 PRIV	CH13 PRIV	CH12 PRIV	CH11 PRIV	CH10 PRIV	CH9 PRIV	CH8 PRIV	CH7 PRIV	CH6 PRIV	CH5 PRIV	CH4 PRIV	CH3 PRIV	CH2 PRIV	CH1 PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnPRIV:** Processor 1 channel n privilege enable (n = 16 to 1).
When processor 1 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.
When processor 1 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded.
1: Channel n processor 1 privilege enabled (privileged)
0: Channel n processor 1 privilege disabled (unprivileged).

21.4.11 IPCC processor 1 CID configuration register (IPCC_C1CIDCFGR)

Address offset: 0x088

Reset value: 0x0000 0000

This register can only be written by the secure, privilege, trusted domain CID. An illegal write access to this register generates a CID illegal access event.

This register has no read access limitations, and can be read by any domain CID, and any access protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[3:0]				Res.	Res.	Res.	CFEN
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CID[3:0]:** Processor 1 CID domain identification
Processor 1 CID domain identification number:
- 0x0: Processor CID 0
- 0x1: Processor CID 1
- 0x2: Processor CID 2
- 0x3: Processor CID 3
- 0x4: Processor CID 4
- 0x5: Processor CID 5
- 0x6: Processor CID 6
- 0x7: Processor CID 7
- Others: Reserved

If CFEN = 1 these bits are used for processor 1 CID filtering and interrupt routing.
If CFEN = 0 these bits are ignored and there is no processor 1 CID filtering nor interrupt routing.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** CID filtering enabled for processor 1
0: disable the processor 1 CID filtering and interrupt routing, Processor 1 interrupts are broadcasted to all CIDs.
1: enable the processor 1 CID filtering and interrupt routing

21.4.12 IPCC processor 2 security configuration register (IPCC_C2SECCFGR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is CID-filtered depending on the CID configuration in the processor 2 IPCC_C2CIDCFGR. When CID filtering is enabled, this register can be accessed only by the enabled CID. An illegal CID write access is discarded and a read access returns zero data, on illegal accesses a CID illegal access event is generated.

This register can be write-accessed only by a secure privileged access. A nonsecure or unprivileged write access is discarded and causes the generation of an illegal access event. All CID filtered read accesses return the register data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 SEC	CH15 SEC	CH14 SEC	CH13 SEC	CH12 SEC	CH11 SEC	CH10 SEC	CH9 SEC	CH8 SEC	CH7 SEC	CH6 SEC	CH5 SEC	CH4 SEC	CH3 SEC	CH2 SEC	CH1 SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnSEC:** Processor 2 channel n security enable (n = 16 to 1)
1: Channel n processor 2 security enabled (secure)
0: Channel n processor 2 security disabled (nonsecure).

21.4.13 IPCC processor 2 privilege configuration register (IPCC_C2PRIVCFGR)

Address offset: 0x094

Reset value: 0x0000 0000

This register is CID-filtered depending on the CID configuration in the processor 2 IPCC_C2CIDCFGR. When CID filtering is enabled this register can be accessed only by the enabled CID. An illegal CID write access is discarded and a read access returns zero data, on illegal accesses a CID illegal access event is generated.

This register can be write-accessed only by a privilege access. An unprivileged write access is discarded and causes the generation of a processor 2 illegal access event. All CID filtered read accesses return the register data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH16 PRIV	CH15 PRIV	CH14 PRIV	CH13 PRIV	CH12 PRIV	CH11 PRIV	CH10 PRIV	CH9 PRIV	CH8 PRIV	CH7 PRIV	CH6 PRIV	CH5 PRIV	CH4 PRIV	CH3 PRIV	CH2 PRIV	CH1 PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CHnPRIV:** Processor 2 channel n privilege enable (n = 16 to 1).

When processor 2 CHnSEC is disabled, bit n can be accessed with secure and nonsecure access.

When processor 2 CHnSEC is enabled, bit n can be accessed only with secure access. Nonsecure write to this bit is discarded.

1: Channel n processor 2 privilege enabled (privileged)

0: Channel n processor 2 privilege disabled (unprivileged).

21.4.14 IPCC processor 2 CID configuration register (IPCC_C2CIDCFGR)

Address offset: 0x098

Reset value: 0x0000 0000

This register can only be written by the secure, privilege, trusted domain CID. An illegal write access to this register generates a CID illegal access event.

This register has no read access limitations, and can be read by any domain CID, and any access protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[3:0]				Res.	Res.	Res.	CFEN
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CID[3:0]:** Processor 2 CID domain identification

Processor 2 CID domain identification number.
- 0x0: Processor CID 0
- 0x1: Processor CID 1
- 0x2: Processor CID 2
- 0x3: Processor CID 3
- 0x4: Processor CID 4
- 0x5: Processor CID 5
- 0x6: Processor CID 6
- 0x7: Processor CID 7
- Others: Reserved

If CFEN = 1 these bits are used for processor 2 CID filtering and interrupt routing.
If CFEN = 0 these bits are ignored and there is no processor 2 CID filtering nor interrupt routing.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** CID filtering enabled for processor 2

0: disable the processor 2 CID filtering and interrupt routing, Processor 2 interrupts are broadcasted to all CIDs.
1: enable the processor 2 CID filtering and interrupt routing.

21.4.15 IPCC hardware configuration register (IPCC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0300 0310

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CIDWIDTH[3:0]				OPTBITS[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SECCTRL[3:0]				CHANNELS[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CIDWIDTH[3:0]:** CID bit field width range 0 to 12.

0: No CID filtering
1..12: CID filtering with CID bit field with 1..12
13..15: Reserved

Bits 23:16 **OPTBITS[7:0]:** Reserved for future use

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **SECCTRL[3:0]:** Security options, range 1 to 16.

0: No security
1: Security
2: Security and privilege
3: Security, privilege and resource isolation.

Bits 7:0 **CHANNELS[7:0]:** Number of channels per CPU supported by the IP, range 1 to 16

21.4.16 IPCC version register (IPCC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** IPCC major revision number

Bits 3:0 **MINREV[3:0]:** IPCC minor revision number.

21.4.17 IPCC identification register (IPCC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0071

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]**: IPCC identification.

21.4.18 IPCC size identification register (IPCC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: IPCC size identification.

21.4.19 IPCC register map

Table 167. IPCC register map and reset values

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	IPCC_C1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECTXFIE	TXFIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECRXOIE	RXOIE
	Reset value															0	0															0	0
0x0004	IPCC_C1MR	CH16FM	CH15FM	CH14FM	CH13FM	CH12FM	CH11FM	CH10FM	CH9FM	CH8FM	CH7FM	CH6FM	CH5FM	CH4FM	CH3FM	CH2FM	CH1FM	CH16OM	CH15OM	CH14OM	CH13OM	CH12OM	CH11OM	CH10OM	CH9OM	CH8OM	CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0008	IPCC_C1SCR	CH16S	CH15S	CH14S	CH13S	CH12S	CH11S	CH10S	CH9S	CH8S	CH7S	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S	CH16C	CH15C	CH14C	CH13C	CH12C	CH11C	CH10C	CH9C	CH8C	CH7C	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	IPCC_C1TOC2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16F	CH15F	CH14F	CH13F	CH12F	CH11F	CH10F	CH9F	CH8F	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	IPCC_C2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECTXFIE	TXFIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECRXOIE	RXOIE
	Reset value															0	0															0	0
0x0014	IPCC_C2MR	CH16FM	CH15FM	CH14FM	CH13FM	CH12FM	CH11FM	CH10FM	CH9FM	CH8FM	CH7FM	CH6FM	CH5FM	CH4FM	CH3FM	CH2FM	CH1FM	CH16OM	CH15OM	CH14OM	CH13OM	CH12OM	CH11OM	CH10OM	CH9OM	CH8OM	CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0018	IPCC_C2SCR	CH16S	CH15S	CH14S	CH13S	CH12S	CH11S	CH10S	CH9S	CH8S	CH7S	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S	CH16C	CH15C	CH14C	CH13C	CH12C	CH11C	CH10C	CH9C	CH8C	CH7C	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	IPCC_C2TOC1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16F	CH15F	CH14F	CH13F	CH12F	CH11F	CH10F	CH9F	CH8F	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0080	IPCC _C1SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16SEC	CH15SEC	CH14SEC	CH13SEC	CH12SEC	CH11SEC	CH10SEC	CH9SEC	CH8SEC	CH7SEC	CH6SEC	CH5SEC	CH4SEC	CH3SEC	CH2SEC	CH1SEC
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	IPCC _C1PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16PRIV	CH15PRIV	CH14PRIV	CH13PRIV	CH12PRIV	CH11PRIV	CH10PRIV	CH9PRIV	CH8PRIV	CH7PRIV	CH6PRIV	CH5PRIV	CH4PRIV	CH3PRIV	CH2PRIV	CH1PRIV
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0088	IPCC_C1CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[3:0]				Res.	Res.	Res.	CFEN
	Reset value																									0	0	0	0				0
0x0090	IPCC _C2SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16SEC	CH15SEC	CH14SEC	CH13SEC	CH12SEC	CH11SEC	CH10SEC	CH9SEC	CH8SEC	CH7SEC	CH6SEC	CH5SEC	CH4SEC	CH3SEC	CH2SEC	CH1SEC
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	IPCC _C2PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH16PRIV	CH15PRIV	CH14PRIV	CH13PRIV	CH12PRIV	CH11PRIV	CH10PRIV	CH9PRIV	CH8PRIV	CH7PRIV	CH6PRIV	CH5PRIV	CH4PRIV	CH3PRIV	CH2PRIV	CH1PRIV
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 167. IPCC register map and reset values (continued)

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0098	IPCC_C2CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[3:0]				Res.	Res.	Res.	CFEN
	Reset value																									0	0	0	0				0
0x03F0	IPCC_HWCFGR	Res.	Res.	Res.	Res.	CIDWIDTH [3:0]				OPTBITS[7:0]								Res.	Res.	Res.	Res.	SECCTRL [3:0]				CHANNELS[7:0]							
	Reset value					0	0	1	1	0	0	0	0	0	0	0	0					0	0	1	1	0	0	1	1	0	0	0	0
0x03F4	IPCC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	0
0x03F8	IPCC_PIDR	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
0x03FC	IPCC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4 on page 184* for the register boundary addresses.

22 General-purpose I/Os (GPIO)

22.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), one 16-bit reset register (GPIOx_BRR), and one 32-bit set/reset register (GPIOx_BSRR).

All GPIOs have one 32-bit locking register (GPIOx_LCKR), two couple of 32-bit advanced configuration registers (GPIOx_DELAYRL/H, GPIOx_ADVCFGRL/H), and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

The access can be restricted to each GPIO configuration bit to secure-only and/or privileged-only. These configuration bits can also be allocated to a specific CPU.

22.2 GPIO main features

- Multiple choice of configurations per I/O port:
  - Input configuration in floating, pull-up/down, or analog state
  - Analog configuration (output buffer and Schmitt trigger inputs disabled)
  - Output configuration, or alternate function configuration, in push-pull or open-drain state, with pull-up or pull-down activated
- Data present on the I/O pin sampled to GPIOx_IDR register (input configuration), or to the peripheral (alternate-function configuration)
- Output buffer on the I/O pin driven by GPIOx_ODR register (output configuration) or by the peripheral (alternate-function configuration)
- I/O data output atomic read/modify through GPIOx_ BSRR and GPIOx_BRR registers
- Speed selection for each I/O (GPIOx_OSPEEDR)
- Lock mechanism (GPIOx_LCKR) to selectively freeze the I/O port configurations
- Highly-flexible pin multiplexing allowing the use of I/O pins as GPIOs, or as one of several possible peripheral function
- Programmable delay to either the input path or the output path (GPIOx_DELAYR)
- Double-edge selection, clock inversion, and optional retime (GPIOx_ADVCFGR)
- Possibility to restrict each I/O control to secure-only, privileged-only, or to a specific CPU (using its compartment information)

22.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the GPIO ports can be individually configured by software in several modes:

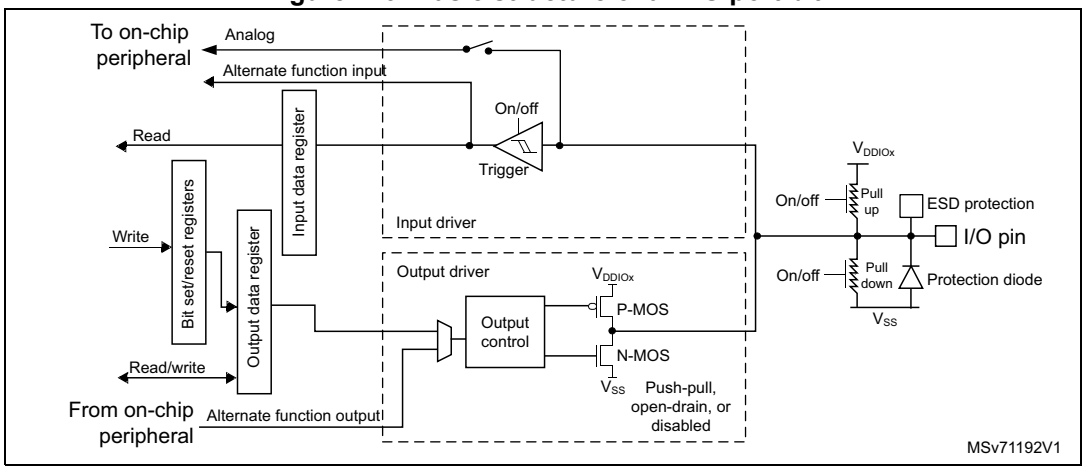
- input floating
- input pull-up
- input-pull-down
- analog
- output-open-drain with pull-up or pull-down capability

- output-push-pull with pull-up or pull-down capability
- alternate-function push-pull with pull-up or pull-down capability
- alternate-function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable. GPIO registers must be accessed as 32-bit words, half-words, or bytes. GPIOx_BSRR and GPIOx_BRR allow atomic read/modify accesses to any of GPIOx_ODR. There is then no risk of an IRQ occurring between the read and the modify access.

The figure below shows the basic structure of a standard I/O port bit. *Table 168* gives the possible port bit configurations.

Figure 129. Basic structure of an I/O port bit



I/Os can be used at 1.8 V typ. or 3.3 V typ. (default), by selecting the VDDIOxRSEL range in PWR_CR1.

A 1.8 V I/O supply voltage and 1.8 V IO levels, with a 3.3 V I/O range setting is allowed, but results in degraded I/O performances.

A 3.3V I/O supply voltage or 3.3 V I/O levels, with a 1.8 V I/O range setting is not allowed, and may permanently damage the I/O.

Table 168. Port x bit configurations

Port x configuration bits					I/O configuration(1)	
MODEx[1:0]	OTx	OSPEEDx[1:0]	PUPDx[1:0]			
01	0	SPEED[1:0]	0	0	GP output	PP
			0	1		PP + PU
			1	0		PP + PD
			1	1	Reserved (GP output PP)	
	1		0	0	GP output (open-drain)	OD
			0	1		OD + PU
			1	0		OD + PD
			1	1	Reserved (GP output OD)	

Table 168. Port x bit configurations (continued)

Port x configuration bits					I/O configuration(1)	
MODEx[1:0]	OTx	OSPEEDx[1:0]	PUPDx[1:0]			
10	0	SPEED[1:0]	0	0	AF	PP
			0	1		PP + PU
			1	0		PP + PD
			1	1	Reserved (AF PP)	
	1		0	0	AF (open-drain)	OD
			0	1		OD + PU
			1	0		OD + PD
			1	1	Reserved (AF OD)	
00	x	x	0	0	GP input	Floating
	x	x	0	1		PU
	x	x	1	0		PD
	x	x	1	1	Reserved (input floating)	
11	x	x	0	0	Input/output	Analog
	x	x	0	1		Analog
	x	x	1	0		Analog + PD
	x	x	1	1		Analog

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

22.3.1 General-purpose I/O (GPIO)

During and just after a reset, alternate functions are not active, and all I/O ports are configured in analog mode, as defined through GPIOx_MODER.

When the pin is configured as output, the value written to GPIOx_ODR is output on the I/O pin. The output driver can be used in push-pull or open-drain mode (only the low level is driven, the high level is Hi-Z).

GPIOx_IDR captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, that can be activated or not depending on GPIOx_PUPDR value.

22.3.2 I/O pin alternate-function multiplexer and mapping

The I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. This avoids any conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15), that can be configured through GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15):

- During and just after reset, AFs are not active, and all I/O ports are configured in analog mode (through GPIOx_MODER). When active, AF0 is selected by default.

- Specific AF assignments for each I/O pin are detailed in the device datasheet.

This flexible I/O multiplexing architecture is used to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- GPIO modes: configure the desired I/O as output, input, or analog in GPIOx_MODER.
- Peripheral alternate function:
  - For a given peripheral function, use the device datasheet “Alternate functions” table to identify the AF to use with a given I/O. One of GPIOx_AFRL or GPIOx_AFRH must be used.
  - Configure the desired I/O as an AF in GPIOx_MODER. Select also the type, pull-up/pull-down, and output speed via GPIOx_OTYPER, GPIOx_PUPDR, and GPIOx_OSPEEDR respectively.
- Cortex-M33 alternate function (EVENTOUT)
  - The Cortex-M33 output EVENTOUT signal can be used by configuring the I/O pin to output at AF15. An event can be signaled through the configured pin after executing SEV instruction.
  - EVENTOUT signal can be used internally as a trigger for some peripherals (see *Section 24: Peripheral interconnects*).
- Additional functions:
  - In the device datasheet, the “Additional functions” table lists peripheral functions that directly select/enable its allocated I/O port(s). Typical peripherals are RTC, TAMP, PWR (WKUPx pins), or RCC (oscillator pins). These functions have priority over the configuration in standard GPIO registers.
  - For ADC peripherals, configure the desired I/O in analog mode in GPIOx_MODER and configure the required function in ADC registers.
  - As described in the device datasheet “Additional functions”, some pins/balls are directly connected to ANA0 and ANA1 ADC analog inputs.

22.3.3 I/O port control registers

Each GPIO port x has four 32-bit memory-mapped control registers to configure up to 16 I/Os:

- GPIOx_MODER is used to select the I/O mode (input, output, AF, analog).
- GPIOx_OTYPER and GPIOx_OSPEEDR are respectively used to select the output type (push-pull or open-drain) and the speed.
- GPIOx_PUPDR is used to select the pull-up/pull-down whatever the I/O direction.

22.3.4 I/O port data registers

Each GPIO port x has two 16-bit memory-mapped data registers:

- GPIOx_ODR read/writable register stores the data to be output per I/O.
- GPIOx_IDR read-only register stores the data input through each I/O.

22.3.5 I/O data bitwise handling

GPIOx_BSRR allows the application to set and reset each individual bit in GPIOx_ODR. GPIOx_BSRR has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using GPIOx_BSRR to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock GPIOx_ODR bits (they can always be accessed directly). GPIOx_BSRR provides a way of performing atomic-bitwise handling.

The software does not need to disable interrupts when programming GPIOx_ODR at bit level: one or more bits can be modified in a single-atomic AHB write access.

22.3.6 GPIO locking mechanism

GPIO control registers can be frozen by applying a specific write sequence to GPIOx_LCKR: each GPIOx_LCKR bit freezes the corresponding bit in GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_DELAYRL/H, GPIOx_ADVCFGRL/H, and GPIOx_AFRL/H.

To write GPIOx_LCKR, a specific write/read sequence must be applied. When the right lock sequence is applied to the bit 16 in this register, LCKR[15:0] value is used to lock the I/O configuration (during the write sequence, LCKR[15:0] value must be the same). When the lock sequence is applied to a port bit, the port bit value can no longer be modified until next device or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_DELAYRL/H, GPIOx_ADVCFGRL/H, and GPIOx_AFRL/H).

The lock sequence can only be performed using a 32-bit word access to GPIOx_LCKR, due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

Note: The lock sequence aborts in the event of an AHB bus error (see Section 22.4 introduction), or insufficient access rights (see Section 22.3.15 to Section 22.3.17).

The lock sequence automatically suspends when accessing a different register from GPIOx_LCKR. It resumes with the next access to GPIOx_LCKR.

22.3.7 I/O alternate-function input/output

GPIOx_AFRL and GPIOx_AFRH registers are used to select one of the AF inputs/outputs available for each I/O. With these registers, the user can connect an AF to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH. The application can thus select any one of the possible functions for each I/O. The AF-selection signal being common to the AF input and AF output, a single channel is selected for the AF input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

22.3.8 External interrupt/wake-up lines

All ports have an external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate-function mode (the port must not be configured

in analog mode). Refer to *Section 29: Extended interrupts and event controller (EXTI)* for more details.

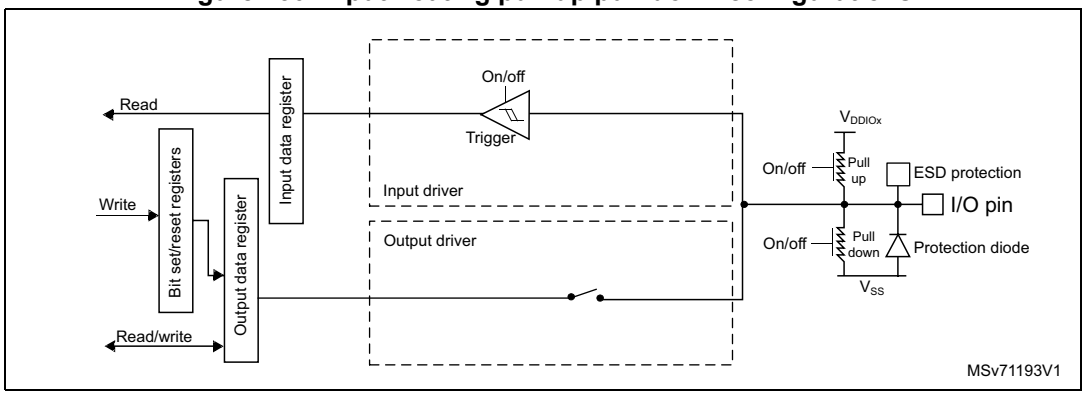
22.3.9 Input configuration

When the I/O port is programmed as input, the following happens:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- Pull-up and pull-down resistors are activated depending on GPIOx_PUPDR value.
- Data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 130. Input floating/pull-up/pull-down configurations



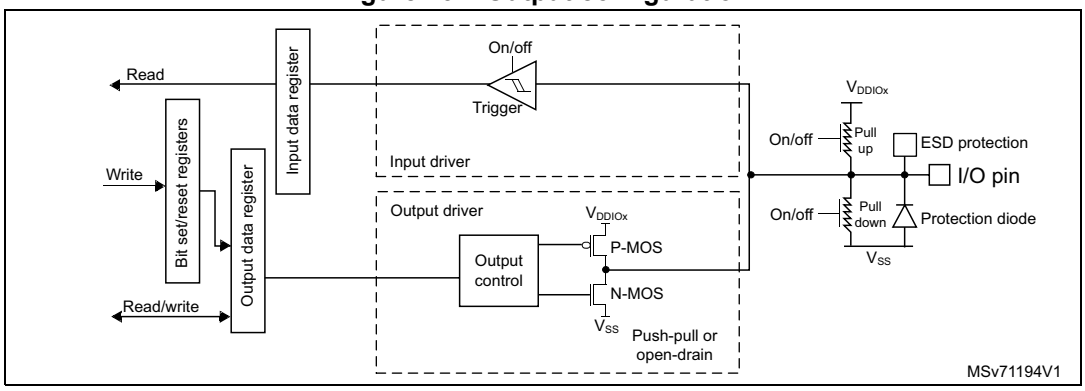
22.3.10 Output configuration

When the I/O port is programmed as output, the following happens:

- The output buffer is enabled:
  - Open-drain mode: a zero in the output register activates the N-MOS. A one in the output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: a zero in the output register activates the N-MOS. A one in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- Pull-up and pull-down resistors are activated depending on GPIOx_PUPDR value.
- Data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 131. Output configuration



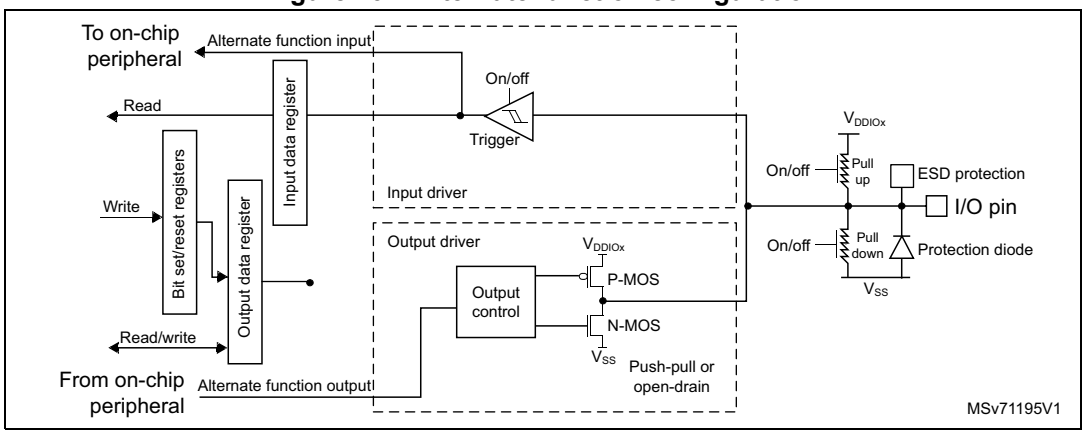
22.3.11 Alternate-function configuration

When the I/O port is programmed as alternate function, the following happens:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- Weak pull-up and pull-down resistors are activated or not depending on GPIOx_PUPDR value.
- Data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 132. Alternate function configuration



22.3.12 Analog configuration

When the I/O port is programmed as analog, the following happens:

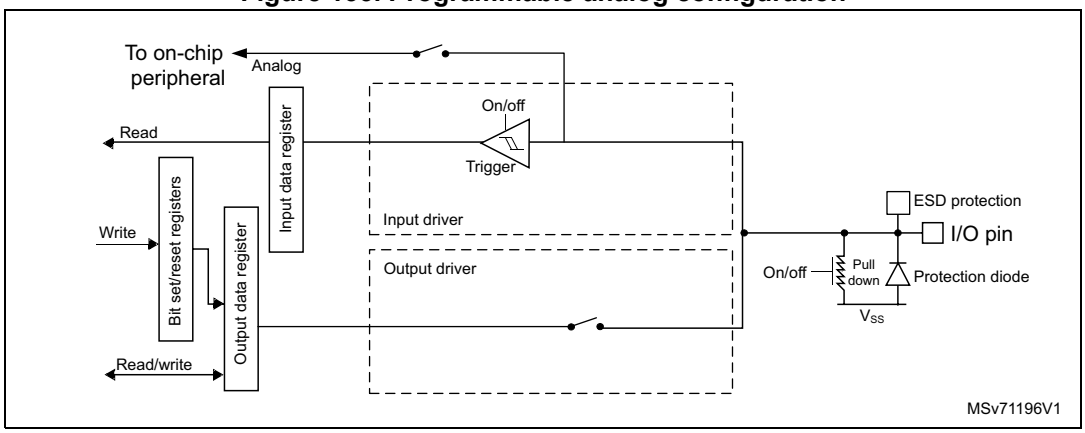
- The output buffer is disabled.

- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up resistor is disabled by hardware. The weak pull-down resistor is activated or not depending on GPIOx_PUPDR value.
- Read access to the input data register gets the value 0.
- The analog switch to the analog peripheral must be closed by control from the analog peripheral.

Caution: The ADC is 1.8 V only. Check the datasheet for allowed voltage on ADC input pins to avoid permanent damage to the ADC.

The figure below shows the programmable, analog-input configuration of the I/O port bits.

Figure 133. Programmable analog configuration



22.3.13 GPIO pins used in V_{SW} domain

The GPIO functionality is lost when the core supply domain is powered off. The pins are then set in analog input mode, unless one of the following conditions is met:

- The GPIO configuration is bypassed by an additional function which stays powered: for example debug, PWR (WKUP only), RTC, TAMP, or RCC (LSE only). For more information, refer to the relevant sections in this reference manual and datasheet.
- The GPIO bank stays powered (for example GPIOZ). Refer to the power supply list of GPIO table in the datasheet for the list of GPIO pins in V_{SW} domain.

22.3.14 Advanced I/O configurations

I/O compensation cell

The I/O commutation slew rate (t_{fall}/t_{rise}) can be adapted by software depending on PVT (process, voltage, and temperature) conditions, in order to reduce the I/O noise on power supply. Refer to *Section 23: System configuration controller (SYSCFG)* for more details.

I/O path configurable delay

A delay programmed in GPIOx_DELAYRL/H can be applied to either input or output path, based on DLYPATH in GPIOx_ADVCFGRL/R.

I/O data double-edge selection

Input and output data can be managed as single- or double-edge, according to DE in GPIOx_ADVCFGRL/R.

I/O data retiming

When RET is set in GPIOx_ADVCFGRL/R, input and output data are retimed to either rising or falling clock edge, depending on INVCLK value.

22.3.15 I/O pin isolation using TrustZone

As TrustZone-aware peripheral, each I/O pin of GPIO port x can be individually configured as secure through GPIOx_SECCFGR. After reset, all GPIO ports are secure.

I/Os connected to peripheral functions can be conditioned by the peripheral security configuration (see *Section 5.4: Resource isolation framework (RIF)* for more details):

- peripherals for which the I/O pin selection is done through GPIOx_AFRL/H

  If the peripheral is configured as secure, it cannot be connected to a non-secure I/O pin. If this is not respected, the input data to the secure peripheral is forced to zero (I/O input pin value ignored), and the output pin value is forced to zero. Any secure information leak through non-secure I/Os is then avoided. TrustZone-aware logic around GPIO ports is summarized in *Table 169*.
- peripherals (like ADC) that embed AFs that directly select/enable its allocated I/O ports using analog switches

  if the I/O is secure, this analog switch cannot be controlled by a non-secure peripheral. If the peripheral is configured as non-secure and the I/O is secure, the switch remains open. This prevents the redirection of secure data to a non-secure peripheral or I/O through analog path. Refer to *Section 5: System security* for the list of peripherals using this security.
- I/Os that do not have an hardware protection linked to TrustZone (see *Section 5: System security*)

  The listed signals (input and/or outputs) are not blocked when the I/O is set as secure. The associated peripheral is non secure. For each of these I/Os, the secure application must decide if a potential effect on data integrity or confidentiality is critical or not.

Refer to the ‘pin definition’ table in the device datasheet for more information about peripheral AFs (first above bullet) and additional-function mapping (last two bullets).

Table 169. Secure AF between peripherals and allocated I/Os

Security configuration		Alternate-function logic		Comment
Peripheral	Allocated I/O pin	Input	Output	
Secure	Secure	I/O data	Peripheral data	-
Non-secure				Out of reset configuration
Secure	Non-secure	Zero	Zero	-
Non-secure		I/O data	Peripheral data	

Table 170 lists the I/O port y secured bits, following the SECy setting in GPIOx_SECCFGR. For each register in the table:

- When a bit is secure, read and write operations are only allowed by a secure access. Non-secure read or write accesses on secure bits are RAZ/WI, with no illegal access event generated.
- When a bit is non-secure, there is no TrustZone restriction. Read and write operations are allowed by both secure and non-secure accesses.

Table 170. GPIO protected bits

Protected bit	Register name	Illegal access to protected bit
MODEy[1:0]	GPIOx_MODER	RAZ/WI
OTy	GPIOx_OTYPER	
OSPEEDy[1:0]	GPIOx_OSPEEDR	
PUPDy[1:0]	GPIOx_PUPDR	
IDy	GPIOx_IDR	
ODy	GPIOx_ODR	
BSy and BRy	GPIOx_BSRR	
BRy	GPIOx_BRR	
LCKy	GPIOx_LCKR	
AFSELy[3:0]	GPIOx_AFRH	
	GPIOx_AFRL	
DELAYy[3:0]	GPIOx_DELAYRL	RAZ/WI
	GPIOx_DELAYRH	
RETy, INVCLKy, DEy, DELYPATHy	GPIOx_ADVCFGRL	
	GPIOx_ADVCFGRH	

Note: GPIOx_SECCFGR is readable by any application, and writable by only secure privileged applications.

Each bit in GPIOx_SECCFGR is write locked until the next device reset when setting the corresponding bit in GPIOx_RCFGLOCKR.

22.3.16 I/O pin isolation using privilege

As RIF-aware peripheral, each I/O pin of GPIO port x can be individually configured as privileged through GPIOx_PRIVCFGR . After reset, all GPIO registers can be read and written by privileged and unprivileged accesses, if the security is enough and the CPU is allowed.

Table 170 in the previous section lists the I/O port y protected bits, following PRIVy setting in GPIOx_PRIVCFGR. For each register in the table:

- When a bit is privileged, read and write operations are only allowed by a privileged access. Unprivileged read or write accesses on privileged bits are RAZ/WI, with no illegal access event generated.
- When a bit is unprivileged, there is no privilege restriction. Read and write operations are allowed by both privileged and unprivileged accesses.

Note: GPIOx_PRIVCFGR is readable by any application, writable by only privileged applications.

Each bit in GPIOx_PRIVCFGR is write locked until the next device reset, when setting the corresponding bit in GPIOx_RCFGLOCKR.

22.3.17 I/O pin isolation using compartments

As RIF-aware peripheral, when the compartment-based filtering is active (CFEN = 1), each I/O pin y of GPIO port x can be individually configured as controlled by a unique CPU using one of the following techniques:

- A trusted-domain application keeps SEM_EN cleared, and sets the authorized compartment with SCID[1:0] in GPIOx_CIDCFGRy.
- A trusted-domain application sets SEM_EN, and sets compartments that are authorized to take the I/O pin y, by setting the corresponding SEMWLCm bits in GPIOx_CIDCFGRy. Each CPU, white-listed this way, is authorized to take SEM_MUTEX in GPIOx_SEMCRy to have the exclusive ownership on the I/O pin until this CPU clears SEM_MUTEX.

Table 170 in *Section 22.3.15* details the I/O port y protected bits following the isolation to a compartment using GPIOx_CIDCFGRy and GPIOx_SEMCRy. For each register in the table:

- For bits with compartment filtering activated: read and write operations are only allowed to the CPU with the correct compartment ID (CID), if secure and privilege of the access are enough. Other CPUs read configuration bits as zero, and writes are ignored. There is no illegal access event generated.
- For bits with compartment filtering deactivated: no restriction. Read and write operations are allowed by any CPU, if secure and privilege of the access are enough.

Note: GPIOx_CIDCFGR and GPIOx_SEMCR are readable by any application. GPIOx_CIDCFGR is writable only by the trusted-domain CPU.

GPIOx_CIDCFGRy is write locked until the next device reset, when setting RLOCKy in GPIOx_RCFGLOCKR.

For more information on RIF, compartment ID (CID) and trusted domain application, refer to *Section 5.4: Resource isolation framework (RIF)*.

22.4 GPIO registers

These registers are accessible through 32-bit word, half-word, or byte single access. Double-word and instruction-fetch accesses trigger an AHB bus error. Byte or half-word writes to GPIOx_LCKR are silently ignored (no bus error).

For any DDCID access to registers, the security and privileged policy applies like any other CIDs. Compartment filtering policy also applies to DDCID access if the debug domain is deactivated (DDEN = 0). When the debug domain is activated (DDEN = 1), the DDCID access goes through regardless of any CID filtering policy.

22.4.1 GPIO port x mode register (GPIOx_MODER) (x = A to I, Z)

Address offset:0x000

Reset value: 0xFFFF FFFF (A, D, E port)

Reset value: 0xFFFC FCFC (B port)

Reset value: 0x0FFF FFFF (C port)

Reset value: 0xCFFF FFFF (F port)

Reset value: 0xFFFF CFFF (G port)

Reset value: 0x0FFF CF00 (H port)

Reset value: 0x0003 3F0F (I port)

Reset value: 0x0000 00CF (Z port)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

00: General-purpose input mode

01: General-purpose output mode

10: Alternate-function mode (refer to device datasheet for available options)

11: Analog mode (reset state)

Successful read or write to MODEy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1, it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.2 GPIO port x output type register (GPIOx_OTYPER) (x = A to I, Z)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bit, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.

If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.3 GPIO port x output speed register (GPIOx_OSPEEDR) (x = A to I, Z)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for frequency specifications, power supply, and load conditions for each speed.

Successful read or write to the bitfield OSPEEDy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I, Z)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Successful read or write to the bitfield PUPDy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.5 GPIO port x input data register (GPIOx_IDR) (x = A to I, Z)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.
Successful read to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read, if privilege and security of the access is sufficient. Illegal read returns zero.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.6 GPIO port x output data register (GPIOx_ODR) (x = A to I, Z)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port x output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to GPIOx_BSRR or GPIOx_BRR.

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bit, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.7 GPIO port x bit set/reset register (GPIOx_BSRR) (x = A to I, Z)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Successful write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can write to this bit, if privilege and security of the access is sufficient. Illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

Successful write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can write to this bit, if privilege and security of the access is sufficient. Illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.8 GPIO port x configuration lock register (GPIOx_LCKR) (x = A to I, Z)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the GPIO configuration. During the lock sequence, LCKR[15:0] value can change, but only stable bits are set (for example: if LCKR[15:0] = 0x3 during the first two writes but LCKR[15:0] = 0x2 for the third write, then only LCK1 is set).

Note: Only word access (32-bit long) is allowed during this locking sequence. Byte or half-word writes are silently ignored (no bus error, no abort to lock sequence).

When the lock sequence has been applied on a port bit, this port bit value can no longer be modified until the next device or peripheral reset. Each lock bit freezes a specific configuration register (control and alternate-function registers).
See *Section 22.3.6* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
0: Port configuration lock key not active
1: Port configuration lock key active. This register is locked until the next device or peripheral reset.
LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:0]
WR LCKR[16] = 0 + LCKR[15:0]
WR LCKR[16] = 1 + LCKR[15:0]
RD LCKR
RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active.)

Note: During the lock key write sequence, LCK[15:0] value must not change.
Any error in the lock sequence aborts the lock.
After the first lock sequence on any bit of the port, any read access on LCKK returns one until the next device or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when LCKK = 0.

0: Port configuration not locked

1: Port configuration locked

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bit, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.9 GPIO port x alternate function low register (GPIOx_AFRL) (x = A to I, Z)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Successful read or write to the bitfield AFSELy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.10 GPIO port x alternate function high register (GPIOx_AFRH) (x = A to I, Z)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)

These bits are written by software to configure alternate function I/Os. See GPIOx_AFRL register for details.

Successful read or write to the bitfield AFSELy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.11 GPIO port x bit reset register (GPIOx_BRR) (x = A to I, Z)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit

Successful write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can write to this bit, if privilege and security of the access is sufficient. Illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.12 GPIO port x secure configuration register (GPIOx_SECCFGR) (x = A to I, Z)

Address offset: 0x030

Reset value: 0x0000 FFFF

This register is used to configure a selected I/O as secure. Non-secure or unprivileged writes to this register are ignored, while any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: I/O pin y of Port x security configuration (y = 15 to 0)
0: The I/O pin y is non-secure. Secure access is also possible.
1: The I/O pin y is secure. Refer to *Table 170* for all corresponding secured bits.
If compartment filtering for this I/O pin is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (resp any compartment) can write this bit, if privilege and security of the access is sufficient.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGR.
If SEM_EN = 1 it is equal to SEMCID if SEM_MUTEX = 1 in GIPOx_SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.
If RLOCKy bit is set in GPIOx_RCFGLOCKR write to this bit is ignored.

22.4.13 GPIO port x privileged configuration register (GPIOx_PRIVCFGR) (x = A to I, Z)

Address offset: 0x034

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, while any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVy**: I/O pin y of port x privilege configuration (y = 15 to 0)

0: The I/O pin y is unprivileged. Privileged access is also possible.

1: The I/O pin y is privileged only. Refer to *Table 170* for all corresponding protected bits.

If corresponding SECy is set in GPIOx_SECCFGR, this bit can only be written by a secure privileged application.

If compartment filtering for this I/O pin is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (resp any compartment) can write this bit, if privilege and security of the access is sufficient.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGR.

If SEM_EN = 1 it is equal to SEMCID if SEM_MUTEX = 1 in GIPOx_SEMCR. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

If RLOCKy bit is set in GPIOx_RCFGLOCKR write to this bit is ignored.

22.4.14 GPIO port x resource configuration lock register (GPIOx_RCFGLOCKR) (x = A to I, Z)

Address offset: 0x038

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain CID.

Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK 15	RLOCK 14	RLOCK 13	RLOCK 12	RLOCK 11	RLOCK 10	RLOCK 9	RLOCK 8	RLOCK 7	RLOCK 6	RLOCK 5	RLOCK 4	RLOCK 3	RLOCK 2	RLOCK 1	RLOCK 0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RLOCKy**: I/O pin y of port x resource lock (y = 15 to 0)

This bit is set to lock this I/O resource. It is cleared by default and once set it cannot be cleared until GPIOx peripheral is reset.

0: Bits SECy in GPIOx_SECCFGR and PRIVy in GPIOx_PRIVCFGR are writable. GPIOx_CIDCFGRy is also writable.

1: Writes to bits SECy in GPIOx_SECCFGR and PRIVy in GPIOx_PRIVCFGR are ignored. Writes to GPIOx_CIDCFGRy are also ignored.

22.4.15 GPIO port x delay low register (GPIOx_DELAYRL) (x = A to I, Z)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLY7[3:0]				DLY6[3:0]				DLY5[3:0]				DLY4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLY3[3:0]				DLY2[3:0]				DLY1[3:0]				DLY0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DLYy[3:0]:** Port x IO pin y delay setup (y = 7 to 0)

Configure delay applied to either the input path (data IN from pad) or the output path (data OUT to pad), depending on the value of the corresponding DLYPATHy in GPIOx_ADVCFGRL.

0000: no delay
0001: delay 0.3 ns
0010: delay 0.5 ns
0011: delay 0.75 ns
0100: delay 1.0 ns
0101: delay 1.25 ns
0110: delay 1.5 ns
0111: delay 1.75 ns
1000: delay 2.0 ns
1001: delay 2.25 ns
1010: delay 2.5 ns
1011: delay 2.75 ns
1100: delay 3.0 ns
1101: delay 3.25 ns
1110 to 1111: delay 3.25 ns (maximum)

Successful read or write to the bitfield DLYy must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.16 GPIO port x delay high register (GPIOx_DELAYRH) (x = A to I, Z)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLY15[3:0]				DLY14[3:0]				DLY13[3:0]				DLY12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLY11[3:0]				DLY10[3:0]				DLY9[3:0]				DLY8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DLYy[3:0]:** Port x IO pin y delay setup (y = 15 to 8)

Configure delay on input or output data related to WC prior to retiming.
See GPIOx_DELAYRL for details.

22.4.17 GPIO port x advanced configuration low register (GPIOx_ADVCFGRL) (x = A to I, Z)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RET7	INV CLK7	DE7	DLY PATH7	RET6	INV CLK6	DE6	DLY PATH6	RET5	INV CLK5	DE5	DLY PATH5	RET4	INV CLK4	DE4	DLY PATH4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET3	INV CLK3	DE3	DLY PATH3	RET2	INV CLK2	DE2	DLY PATH2	RET1	INV CLK1	DE1	DLY PATH1	RET0	INV CLK0	DE0	DLY PATH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31, 27, 23, 19, 15, 11, 7, 3 **RETy**: Data retime selection y (y = 7 to 0)

0: Input and output data are not synchronized or retimed on clock edges.

1: Input and output data are retimed to either rising or falling clock edge depending on value of INVCLKy.

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.

If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 30, 26, 22, 18, 14, 10, 6, 2 **INVCLKy**: Clock inversion selection y (y = 7 to 0)

0: I/O clocks are not inverted. Input and output data are retimed to rising clock edge.

1: I/O clocks are inverted. Input and output data are retimed to falling clock edge.

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.

If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 29, 25, 21, 17, 13, 9, 5, 1 **DEy**: Input/Output data double edge selection y (y = 7 to 0)

0: Input and output data is single-edge (changing on rising or falling clock edge, but not both)

1: Input and output data is double-edge (changing on both rising and falling clock edges)

Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.

If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.

Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.

If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 28, 24, 20, 16, 12, 8, 4, 0 **DLYPATHy**: Controls which path contains the configurable delay, input or output y (y = 7 to 0)
0: Delay is switched into the output path, while the input path is set as pass through.
1: Delay is switched into the input path, while the output path is set as pass through.
Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.18 GPIO port x advanced configuration high register (GPIOx_ADVCFGRH) (x = A to I, Z)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RET15	INV CLK15	DE15	DLY PATH 15	RET14	INV CLK14	DE14	DLY PATH 14	RET13	INV CLK13	DE13	DLY PATH1 3	RET12	INV CLK12	DE12	DLY PATH1 2
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET11	INV CLK11	DE11	DLY PATH1 1	RET10	INV CLK10	DE10	DLY PATH1 0	RET9	INV CLK9	DE9	DLY PATH9	RET8	INV CLK8	DE8	DLY PATH8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31, 27, 23, 19, 15, 11, 7, 3 **RETy**: Data retime selection y (y = 15 to 8)
0: Input and output data are not synchronized or retimed on clock edges.
1: Input and output data are retimed to either rising or falling clock edge depending on value of INVCLKy.
Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 30, 26, 22, 18, 14, 10, 6, 2 **INVCLKy**: Clock inversion selection y (y = 15 to 8)
0: I/O clocks are not inverted. Input and output data are retimed to rising clock edge.
1: I/O clocks are inverted. Input and output data are retimed to falling clock edge.
Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 29, 25, 21, 17, 13, 9, 5, 1 **DEy**: Input/Output data double edge selection y (y = 15 to 8)
0: Input and output data is single-edge (changing on rising or falling clock edge, but not both)
1: Input and output data is double-edge (changing on both rising and falling clock edges)
Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

Bits 28, 24, 20, 16, 12, 8, 4, 0 **DLYPATHy**: Controls which path contains the configurable delay, input or output y (y = 15 to 8)
0: Delay is switched into the output path, while the input path is set as pass through.
1: Delay is switched into the input path, while the output path is set as pass through.
Successful read or write to this bit must be secure if SECy is set in GPIOx_SECCFGR, and must be privileged if PRIVy is set in GPIOx_PRIVCFGR.
If compartment filtering for this I/O pin y is enabled (respectively disabled) using CFEN bit in GPIOx_CIDCFGRy, only an authorized compartment (respectively any compartment) can read or write this bitfield, if privilege and security of the access is sufficient. Illegal read returns zero and illegal write is ignored.
Authorized compartment is defined as SCID if SEM_EN = 0 in GPIOx_CIDCFGRy.
If SEM_EN = 1 it is defined as SEMCID, if SEM_MUTEX = 1 in GPIOx_SEMCRy. There is no authorized compartment if SEM_EN = 1 and SEM_MUTEX = 0.

22.4.19 GPIO port x CID configuration register for I/O pin y (GPIOx_CIDCFGRy) (x = A to I, Z)

Address offset: 0x050 + 0x8 * y (y = 0 to 15)

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain CID.

Any read access is allowed on this register. Any write is ignored if RLOCKy is set in GPIOx_RCFGLOCKR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[1:0]		Res.	Res.	SEM_EN	CFEN
										rw	rw			rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **SEMWLCz**: Semaphore white list for compartment z (z = 3 to 0)

0: Application running in compartment z cannot get access to the I/O pin y through semaphore SEM_MUTEX.

1: Application running in compartment z can get access to the I/O pin y through semaphore SEM_MUTEX.

Each bit is not taken into account by GPIOx when semaphore is disabled (SEM_EN = 0), or compartment filtering is off (CFEN = 0) for this I/O pin.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **SCID[1:0]**: Static CID

Trusted domain application uses this bitfield to statically define which compartment has access to the I/O pin y when SEM_EN = 0.

This CID is not taken into account by GPIOx when semaphore is enabled (SEM_EN = 1) or compartment filtering is off (CFEN=0) for this I/O pin.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEM_EN**: Semaphore enable

0: Access to I/O pin y is granted only if CID of the access equals to SCID.

1: Access to I/O pin y is granted only if SEM_MUTEX = 1 and CID of the access equals to SEMCID in GPIOx_SEMCR.

This bit is not taken into account by GPIOx if compartment filtering is off (CFEN = 0) for this I/O pin. When this bit is cleared from 1 by trusted domain application SEM_MUTEX is cleared to zero by GPIOx.

Bit 0 **CFEN**: CID filtering enable

0: Compartment-based filtering defined in GPIOx_CIDCFGR, and optionally GPIOx_SEMCR is disabled for I/O pin y.

1: Compartment-based filtering defined in GPIOx_CIDCFGR, and optionally GPIOx_SEMCR is enabled for I/O pin y.

For any CFEN value access controls to I/O pin y defined in GPIOx_PRIVCFGR and GPIOx_SECCFGR always apply. When this bit is cleared from 1 by trusted domain application SEM_MUTEX is cleared to zero by GPIOx.

22.4.20 GPIO port x semaphore control register for I/O pin y (GPIOx_SEMCRy) (x = A to I, Z)

Address offset: 0x054 + 0x8 * y (y = 0 to 15)

Reset value: 0x0000 0000

If compartment filtering is disabled (CFEN = 0) or semaphore is disabled (SEM_EN = 0) in GPIOx_CIDCFGRy, any write to this register is ignored.

Privileged-only write if PRIVy is set in GPIOx_PRIVCFGR. Secure-only write if SECy is set in GPIOx_SECCFGR.

If compartment filtering is enabled and semaphore is enabled (CFEN and SEM_EN set in GPIOx_CIDCFGRy), only an authorized compartment can write to this register. For this register, the authorized compartment is equal to SEMCID if SEM_MUTEX = 1, else it is defined by the CID white list (SEMWLCm) in GPIOx_CIDCFGRy. There is no authorized compartment if semaphore is disabled (SEM_EN = 0).

Any read access is allowed to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[1:0]		Res.	Res.	Res.	SEM_ MUTEX
										r	r				rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SEMCID[1:0]**: Semaphore current CID

When SEM_EN and CFEN are set in GPIOx_CIDCFGRy, this read-only bitfield is updated with the CID value of the latest successful acquisition of GPIOx_SEMCRy semaphore (SEM_MUTEX = 1). In all other cases SEMCID stays unchanged.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX**: Semaphore mutex

White listed application can set semaphore bit to get exclusive access to the I/O pin y configuration when SEM_EN is set in GPIOx_CIDCFGRy (writes are ignored otherwise).

Read 0: Semaphore is free to take

Read 1: Semaphore is taken

Write 0: Release semaphore

Write 1: Acquire semaphore

When SEM_MUTEX = 0, only applications belonging to white-listed compartments (defined with SEMWLCm bits in GPIOx_CIDCFGRy) can acquire this semaphore.

When SEM_MUTEX = 1, only applications belonging to compartment SEMCID can release this semaphore.

This bit is cleared when CFEN or SEM_EN are cleared from 1 by trusted domain application in GPIOx_CIDCFGRy.

22.4.21 GPIO port x hardware configuration register 11 (GPIOx_HWCFGR11) (x = A to I, Z)

Address offset: 0x3C8

Reset value: 0x0101 1240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PRIV_CFG[3:0]				OR_CFG[3:0]				SEC_CFG[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK_CFG[3:0]				SPEED_CFG[3:0]				AFSIZE_CFG[3:0]				AHB_IOP[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **PRIV_CFG[3:0]:** Privilege mechanism activation
0: Non active
1: Active

Bits 23:20 **OR_CFG[3:0]:** Option register configuration
0: Non active
1: Active

Bits 19:16 **SEC_CFG[3:0]:** Security activation
0: Non active
1: Active

Bits 15:12 **LOCK_CFG[3:0]:** Lock mechanism activation
0: Non active
1: Active

Bits 11:8 **SPEED_CFG[3:0]:** Number of speed lines for each I/O
1: One line
2: Two lines

Bits 7:4 **AFSIZE_CFG[3:0]:** Number of AF available for each I/O
Accepted value: 1 to 4
1: One AF/IO
2: Four AF/IO
3: Eight AF/IO
4: Sixteen AF/IO

Bits 3:0 **AHB_IOP[3:0]:** Bus interface selection
0: AHB
1: IOP

22.4.22 GPIO port x hardware configuration register 10 (GPIOx_HWCFGR10) (x = A to I, Z)

Address offset: 0x3CC

Reset value: 0x0000 FFFF (A, D, E port)

Reset value: 0x0000 FEEE (B port)

Reset value: 0x0000 FFFF (C port)

Reset value: 0x0000 BFFF (F port)

Reset value: 0x0000 FFBF (G port)

Reset value: 0x0000 3FB3 (H port)

Reset value: 0x0000 0173 (I port)

Reset value: 0x0000 000B (Z port)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_IO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EN_IO[15:0]:** Presence granularity, each bit indicate the I/O presence
0: I/O not present
1: I/O present

22.4.23 GPIO port x hardware configuration register 9 (GPIOx_HWCFGR9) (x = A to I, Z)

Address offset: 0x3D0

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FAST_AF_IO15[3:0]				FAST_AF_IO14[3:0]				FAST_AF_IO13[3:0]				FAST_AF_IO12[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAST_AF_IO11[3:0]				FAST_AF_IO10[3:0]				FAST_AF_IO9[3:0]				FAST_AF_IO8[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **FAST_AF_IO15[3:0]:** Indicate which is the fastest AF for I/O15 (0 to F)

Bits 27:24 **FAST_AF_IO14[3:0]:** Indicate which is the fastest AF for I/O14 (0 to F)

Bits 23:20 **FAST_AF_IO13[3:0]:** Indicate which is the fastest AF for I/O13 (0 to F)

Bits 19:16 **FAST_AF_IO12[3:0]:** Indicate which is the fastest AF for I/O12 (0 to F)

Bits 15:12 **FAST_AF_IO11[3:0]:** Indicate which is the fastest AF for I/O11 (0 to F)

Bits 11:8 **FAST_AF_IO10[3:0]:** Indicate which is the fastest AF for I/O10 (0 to F)

Bits 7:4 **FAST_AF_IO9[3:0]:** Indicate which is the fastest AF for I/O9 (0 to F)

Bits 3:0 **FAST_AF_IO8[3:0]:** Indicate which is the fastest AF for I/O8 (0 to F)

22.4.24 GPIO port x hardware configuration register 8 (GPIOx_HWCFGR8) (x = A to I, Z)

Address offset: 0x3D4

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FAST_AF_IO7[3:0]				FAST_AF_IO6[3:0]				FAST_AF_IO5[3:0]				FAST_AF_IO4[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAST_AF_IO3[3:0]				FAST_AF_IO2[3:0]				FAST_AF_IO1[3:0]				FAST_AF_IO0[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **FAST_AF_IO7[3:0]:** Indicate which is the fastest AF for I/O7 (0 to F)

Bits 27:24 **FAST_AF_IO6[3:0]:** Indicate which is the fastest AF for I/O6 (0 to F)

Bits 23:20 **FAST_AF_IO5[3:0]:** Indicate which is the fastest AF for I/O5 (0 to F)

Bits 19:16 **FAST_AF_IO4[3:0]:** Indicate which is the fastest AF for I/O4 (0 to F)

Bits 15:12 **FAST_AF_IO3[3:0]:** Indicate which is the fastest AF for I/O3 (0 to F)

Bits 11:8 **FAST_AF_IO2[3:0]:** Indicate which is the fastest AF for I/O2 (0 to F)

Bits 7:4 **FAST_AF_IO1[3:0]:** Indicate which is the fastest AF for I/O1 (0 to F)

Bits 3:0 **FAST_AF_IO0[3:0]:** Indicate which is the fastest AF for I/O0 (0 to F)

22.4.25 GPIO port x hardware configuration register 7 (GPIOx_HWCFGR7) (x = A to I, Z)

Address offset: 0x3D8

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MODER_RES[31:0]:** MODER register reset value

Valid for bits that are enabled (GPIOx_MODER reset value = MODER_RES[31:0] & GPIOx_HWCFGR10[31:0]).

22.4.26 GPIO port x hardware configuration register 6 (GPIOx_HWCFGR6) (x = A to I, Z)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PUPDR_RES[31:0]:** Pull-up/pull-down register reset value

22.4.27 GPIO port x hardware configuration register 5 (GPIOx_HWCFGR5) (x = A to I, Z)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **OSPEED_RES[31:0]:** OSPEED register reset value

22.4.28 GPIO port x hardware configuration register 4 (GPIOx_HWCFGR4) (x = A to I, Z)

Address offset: 0x3E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OTYPER_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OTYPER_RES[15:0]:** Output type register reset value

Bits 15:0 **ODR_RES[15:0]:** Output data register reset value

22.4.29 GPIO port x hardware configuration register 3 (GPIOx_HWCFGR3) (x = A to I, Z)

Address offset: 0x3E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AFRL_RES[31:0]:** AF register low reset value

22.4.30 GPIO port x hardware configuration register 2 (GPIOx_HWCFGR2) (x = A to I, Z)

Address offset: 0x3EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AFRH_RES[31:0]:** AF register high reset value

22.4.31 GPIO port x hardware configuration register 1 (GPIOx_HWCFGR1) (x = A to I, Z)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OR_RES[15:0]:** Option register reset value

22.4.32 GPIO port x version register (GPIOx_VERR) (x = A to I, Z)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** GPIO major revision
Major revision is 1.

Bits 3:0 **MINREV[3:0]:** GPIO minor revision
Minor revision is 0.

22.4.33 GPIO port x identification register (GPIOx_IPIDR) (x = A to I, Z)

Address offset: 0x3F8

Reset value: 0x000F 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]**: GPIO identifier

22.4.34 GPIO port x size identification register (GPIOx_SIDR) (x = A to I, Z)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier
Bits [31:8] = 0xA3C5DD (fixed code)
Bits [7:0] = 0x01 (1-Kbyte address decoding)

22.4.35 GPIO register map

Table 171. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**GPIOx_MODER (x = A to I, Z)**	MODE15 [1:0]		MODE14 [1:0]		MODE13 [1:0]		MODE12 [1:0]		MODE11 [1:0]		MODE10 [1:0]		MODE9 [1:0]		MODE8 [1:0]		MODE7 [1:0]		MODE6 [1:0]		MODE5 [1:0]		MODE4 [1:0]		MODE3 [1:0]		MODE2 [1:0]		MODE1 [1:0]		MODE0 [1:0]	
	Reset value (A, D, E port)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (B port)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
	Reset value (C port)	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (F port)	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (G port)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (H port)	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0
	Reset value (I port)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1
	Reset value (Z port)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1
0x004	**GPIOx_OTYPER (x = A to I, Z)**	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**GPIOx_OSPEEDR (x = A to I, Z)**	OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]		OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**GPIOx_PUPDR (x = A to I, Z)**	PUPD15 [1:0]		PUPD14 [1:0]		PUPD13 [1:0]		PUPD12 [1:0]		PUPD11 [1:0]		PUPD10 [1:0]		PUPD9 [1:0]		PUPD8 [1:0]		PUPD7 [1:0]		PUPD6 [1:0]		PUPD5 [1:0]		PUPD4 [1:0]		PUPD3 [1:0]		PUPD2 [1:0]		PUPD1 [1:0]		PUPD0 [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**GPIOx_IDR (x = A to I, Z)**	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x014	**GPIOx_ODR (x = A to I, Z)**	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**GPIOx_BSRR (x = A to I, Z)**	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**GPIOx_LCKR (x = A to I, Z)**	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**GPIOx_AFRL (x = A to I, Z)**	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**GPIOx_AFRH (x = A to I, Z)**	AFRH15 [3:0]				AFRH14 [3:0]				AFRH13 [3:0]				AFRH12 [3:0]				AFRH11 [3:0]				AFRH10 [3:0]				AFRH9[3:0]				AFRH8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**GPIOx_BRR (x = A to I, Z)**	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															

Table 171. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	**GPIOx_SECCFGR (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x034	**GPIOx_PRIVCFGR (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**GPIOx_RCFGLOCKR (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLOCK15	RLOCK14	RLOCK13	RLOCK12	RLOCK11	RLOCK10	RLOCK9	RLOCK8	RLOCK7	RLOCK6	RLOCK5	RLOCK4	RLOCK3	RLOCK2	RLOCK1	RLOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	**GPIOx_DELAYRL (x = A to I, Z)**	DLY7[3:0]				DLY6[3:0]				DLY5[3:0]				DLY4[3:0]				DLY3[3:0]				DLY2[3:0]				DLY1[3:0]				DLY0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	**GPIOx_DELAYRH (x = A to I, Z)**	DLY15 [3:0]				DLY14 [3:0]				DLY13 [3:0]				DLY12 [3:0]				DLY11 [3:0]				DLY10 [3:0]				DLY9[3:0]				DLY8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	**GPIOx_ADVCFGRL (x = A to K, Z)**	RET7	INVCLK7	DE7	DLYPATH7	RET6	INVCLK6	DE6	DLYPATH6	RET5	INVCLK5	DE5	DLYPATH5	RET4	INVCLK4	DE4	DLYPATH4	RET3	INVCLK3	DE3	DLYPATH3	RET2	INVCLK2	DE2	DLYPATH2	RET1	INVCLK1	DE1	DLYPATH1	RET0	INVCLK0	DE0	DLYPATH0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	**GPIOx_ ADVCFGRH (x = A to K, Z)**	RET15	INVCLK15	DE15	DLYPATH15	RET14	INVCLK14	DE14	DLYPATH14	RET13	INVCLK13	DE13	DLYPATH13	RET12	INVCLK12	DE12	DLYPATH12	RET11	INVCLK11	DE11	DLYPATH11	RET10	INVCLK10	DE10	DLYPATH10	RET9	INVCLK9	DE9	DLYPATH9	RET8	INVCLK8	DE8	DLYPATH8
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050 + 0x8 * y (y=0 to 15)	**GPIOx_CIDCFGRy (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[1:0]		Res.	Res.	SEM_EN	CFEN
	Reset value													0	0	0	0											0	0			0	0
0x054 + 0x8 * y (y=0 to 15)	**GPIOx_SEMCRy (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[1:0]		Res.	Res.	Res.	SEM_MUTEX
	Reset value																											0	0				0
0xCC-0x3C4	Reserved	Reserved																															
0x3C8	**GPIOx_HWCFGR11 (x = A to I, Z)**	Res.	Res.	Res.	Res.	PRIV_CFG [3:0]				OR_CFG [3:0]				SEC_CFG [3:0]				LOCK_CFG [3:0]				SPEED_CFG [3:0]				AFSIZE_CFG [3:0]				AHB_IOP [3:0]			
	Reset value					0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
0x3CC	**GPIOx_HWCFGR10 (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_IO[15:0]															
	Reset value (A, D, E port)																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (B port)																	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0
	Reset value (C port)																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (F port)																	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (G port)																	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	Reset value (H port)																	0	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1
	Reset value (I port)																	0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1
	Reset value (Z port)																	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Table 171. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3D0	**GPIOx_HWCFGR9 (x = A to I, Z)**	FAST_AF_IO15[3:0]				FAST_AF_IO14[3:0]				FAST_AF_IO13[3:0]				FAST_AF_IO12[3:0]				FAST_AF_IO11[3:0]				FAST_AF_IO10 [3:0]				FAST_AF_IO9 [3:0]				FAST_AF_IO8[3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3D4	**GPIOx_HWCFGR8 (x = A to I, Z)**	FAST_AF_IO7[3:0]				FAST_AF_IO6[3:0]				FAST_AF_IO5[3:0]				FAST_AF_IO4[3:0]				FAST_AF_IO3[3:0]				FAST_AF_IO2[3:0]				FAST_AF_IO1[3:]				FAST_AF_IO0[3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3D8	**GPIOx_HWCFGR7 (x = A to I, Z)**	MODR_RES[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3DC	**GPIOx_HWCFGR6 (x = A to I, Z)**	PUPDR_RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E0	**GPIOx_HWCFGR5 (x = A to I, Z)**	OSPEED_RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E4	**GPIOx_HWCFGR4 (x = A to I, Z)**	OTYPER_RES[15:0]																ODR_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E8	**GPIOx_HWCFGR3 (x = A to I, Z)**	AFRL_RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	**GPIOx_HWCFGR2 (x = A to I, Z)**	AFRH_RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3F0	**GPIOx_HWCFGR1 (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_RES[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3F4	**GPIO_VERR (x = A to I, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	0	0
0x3F8	**GPIO_IPIDR (x = A to I, Z)**	IPID[31:16]																IPID[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x3FC	**GPIO_SIDR (x = A to I, Z)**	SID[31:16]																SID[15:0]															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

23 System configuration controller (SYSCFG)

STM32MP21xx devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Managing OCTOSPI1, SDMMC1, SDMMC2, and SDMMC3 delay block
- Managing system related settings
- Managing I/O compensation cells
- Managing ETH1 and ETH2 clock source selection
- Managing the selection of reference clock frequency, the control of the battery charging procedure, and the trimming of USB2PHY1/USB2PHY2 electrical parameters

23.1 SYSCFG functional description

23.1.1 I/O compensation cell

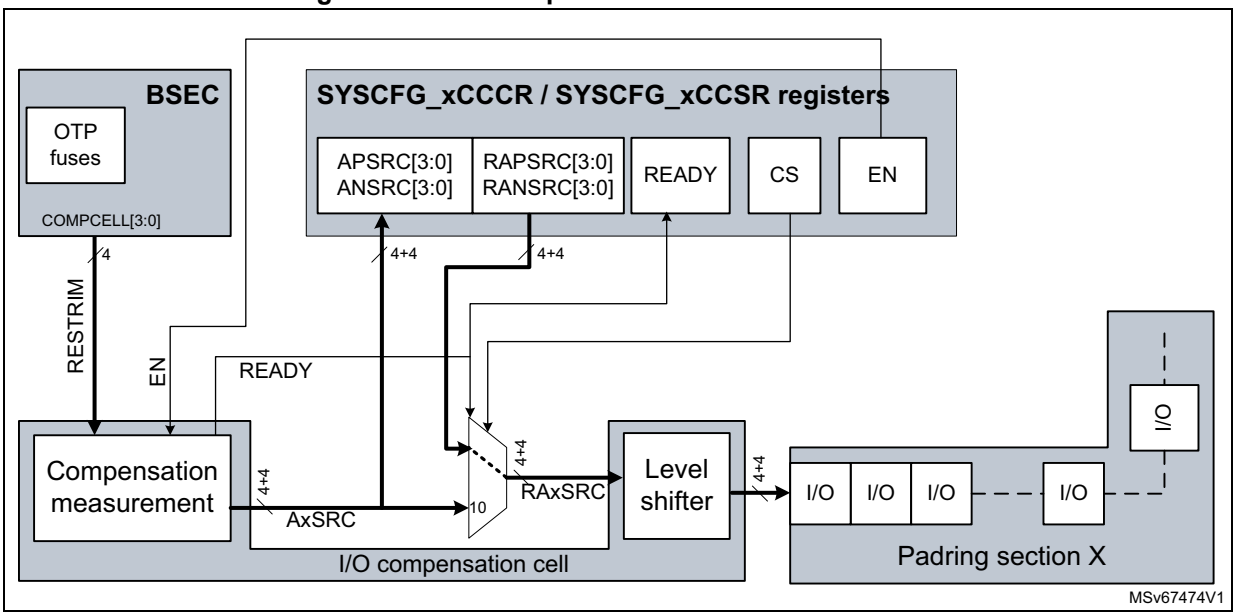
The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the current slew-rate and output impedance in the I/O buffer. Four compensation cells are embedded: one for the I/Os supplied by V_{DD} and three for the I/Os supplied by V_{DDIO1}, V_{DDIO2} and V_{DDIO3}.

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os. When enabled, the compensation cell tracks the PVT, and the 8-bit code APSRC and ANSRC are available in SYSCFG_xCCSR (with x = V_{DD}, V_{DDIO1}, V_{DDIO2} or V_{DDIO3}) once the corresponding READY bit is set. If CS is cleared in SYSCFG_xCCCR and READY is set in SYSCFG_xCCSR, the I/Os receive the code APSRC and ANSRC from SYSCFG_xCCSR, resulting from the compensation cell.

To optimize the trimming, the code can be adjusted using RAPSRC and RANSRC fields in SYSCFG_xCCCR. They can be selected through CS in SYSCFG_xCCCR (see *Figure 134*).

To reduce the power consumption, it is recommended to copy the code from SYSCFG_xCCSR to SYSCFG_xCCCR. After the result is ready, set CS and disable the compensation cell.

Figure 134. I/O compensation cell control overview



23.2 SYSCFG registers

23.2.1 SYSCFG boot pin status register (SYSCFG_BOOTSR)

Address offset: 0x000

Reset value: 0x0000 000X

This register is used to know the state of BOOT pins. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT3	BOOT2	BOOT1	BOOT0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BOOT3**: BOOT3 pin value
0: (low) BOOT3 pin connected to VSS (or left open if BOOT3_PD = 0)
1: (high) BOOT3 pin connected to VDD

Bit 2 **BOOT2**: BOOT2 pin value
0: (low) BOOT2 pin connected to VSS (or left open if BOOT2_PD = 0)
1: (high) BOOT2 pin connected to VDD

Bit 1 **BOOT1**: BOOT1 pin value
0: (low) BOOT1 pin connected to VSS (or left open if BOOT1_PD = 0)
1: (high) BOOT1 pin connected to VDD

Bit 0 **BOOT0**: BOOT0 pin value
0: (low) BOOT0 pin connected to VSS (or left open if BOOT0_PD = 0)
1: (high) BOOT0 pin connected to VDD

23.2.2 SYSCFG boot pin control register (SYSCFG_BOOTCR)

Address offset: 0x004

Reset value: 0x0000 0000

Reset by any system reset.
This register is used to control pull-up to reduce the static power consumption on the pin set to high level.
Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT3_PD	BOOT2_PD	BOOT1_PD	BOOT0_PD
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BOOT3_PD**: BOOT3 pin pull-down disable
This bit is used to save power in case the BOOT3 pin is connected to VDD.
0: (enabled) pull-down enabled. BOOT3 pin can be left open and takes a value of 0 if open.
1: (disabled) pull-down disabled. BOOT3 pin must not be left open.

Bit 2 **BOOT2_PD**: BOOT2 pin pull-down disable
This bit is used to save power in case the BOOT2 pin is connected to VDD.
0: (enabled) pull-down enabled. BOOT2 pin can be left open and takes a value of 0 if open.
1: (disabled) pull-down disabled. BOOT2 pin must not be left open.

Bit 1 **BOOT1_PD**: BOOT1 pin pull-down disable
This bit is used to save power in case the BOOT1 pin is connected to VDD.
0: (enabled) pull-down enabled. BOOT1 pin can be left open and takes a value of 0 if open.
1: (disabled) pull-down disabled. BOOT1 pin must not be left open.

Bit 0 **BOOT0_PD**: BOOT0 pin pull-down disable
This bit is used to save power in case the BOOT0 pin is connected to VDD.
0: (enabled) pull-down enabled. BOOT0 pin can be left open and takes a value of 0 if open.
1: (disabled) pull-down disabled. BOOT0 pin must not be left open.

23.2.3 SYSCFG DLYBSD1 control register (SYSCFG_DLYBSD1CR)

Address offset: 0x400

Reset value: 0x0040 0000

Reset by system reset and SDMMC1 DLL software reset.

Write access according to SDMMC1 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL[5:0]						EN
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **ANTIGLITCH_EN**:
1: The anti-glitch logic is enabled. This ensures that there is no glitch on the clock line when the tap value is modified.

Bits 21:17 **BYP_CMD[4:0]**: Delay unit when in DLL bypass mode
00000: Reserved
00001: 1 delay element enabled
00010: 2 delay elements enabled
...
11000: all 24 delay elements enabled
11001: all 24 delay elements enabled
...
11111: all 24 delay elements enabled

Bit 16 **BYP_EN**: DLL bypass mode enable
1: DLL bypass mode is enabled. In bypass mode the DLL is used in pure delay mode and there is no VT compensation. The command applied to every delay unit is set by DLL_BYP_CMD[4:0], and a tap value must be selected with RX_TAP_SEL[5:0]. In bypass mode, the total delay is given by (delay unit) × tap number selected.
0: DLL lock mode is selected.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:1 **RX_TAP_SEL[5:0]**: TAP number for RX clock selection
000000: 0 tap selected
000001: 1 tap is selected
....
000010: 2 taps selected
....
011111: 31 taps selected
100000: 32 taps selected

Bit 0 **EN**:
When 1, the DLL is enabled when configured in lock mode (BYP_EN = 0).

23.2.4 SYSCFG DLYBSD1 status register (SYSCFG_DLYBSD1SR)

Address offset: 0x404

Reset value: 0x0000 0002

Reset by system reset and SDMMC1 DLL software reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_TAP_SEL_ACK**:
1: The RX delay switched to the tap value specified in RX_TAP_SEL[5:0].

Bit 0 **LOCK**:
1: The DLL is locked.

23.2.5 SYSCFG SDMMC1 control register (SYSCFG_SDMMC1CR)

Address offset: 0x408

Reset value: 0x0000 0000

Reset by system reset and SDMMC1 software reset.

Write access according to SDMMC1 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1_VSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SDMMC1_VSEL**: Voltage selection

Change the value of the SDVSEL1 pin output to control the voltage on pins for external card. Need AFMUX selection.

1: Change the voltage to low-voltage range (1.8 V typ).

0: Change the voltage to high-voltage range (3.3 V typ).

Note: There must be a control circuit out of SDMMC1 to change the voltage on pins.

23.2.6 SYSCFG SDMMC1 address remapping control register (SYSCFG_SDMMC1ARCR)

Address offset: 0x40C

Reset value: 0x0000 0000

Reset by system reset.

Write access according to SDMMC1 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AREN**: Address remapping enable

Enable the address remapping logic in front of the SDMMC1 AHB master. Remapping allows SDMMC1 (only 32bit addressing supported by the master) to access the 4GB of the DDR.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.7 SYSCFG DLYBSD2 control register (SYSCFG_DLYBSD2CR)

Address offset: 0x800

Reset value: 0x0040 0000

Reset by system reset and SDMMC2 DLL software reset.

Write access according to SDMMC2 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL[5:0]						EN
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **ANTIGLITCH_EN**: Anti-glitch logic enable

1: The anti-glitch logic is enabled. This ensures that there is no glitch on the clock line when the tap value is modified.

Bits 21:17 **BYP_CMD[4:0]**: Delay unit when in DLL bypass mode
00000: Reserved
00001: 1 delay element enabled
00010: 2 delay elements enabled
...
11000: all 24 delay elements enabled
11001: all 24 delay elements enabled
...
11111: all 24 delay elements enabled

Bit 16 **BYP_EN**: DLL bypass mode enable
1: DLL bypass mode is enabled. In bypass mode, the DLL is used in pure delay mode and there is no VT compensation. The command applied to every delay unit is set by DLL_BYP_CMD[4:0], and a tap value must be selected with RX_TAP_SEL[5:0]. In bypass mode, the total delay is given by (delay unit) x tap number selected.
0: DLL lock mode is selected.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:1 **RX_TAP_SEL[5:0]**: TAP delay for RX clock selection
000000: 0 tap selected
000001: 1 tap selected
....
000010: 2 taps selected
....
011111: 31 taps selected
100000: 32 taps selected

Bit 0 **EN**:
1: The DLL is enabled when configured in lock mode (BYP_EN = 0).

23.2.8 SYSCFG DLYBSD2 status register (SYSCFG_DLYBSD2SR)

Address offset: 0x804

Reset value: 0x0000 0002

Reset by system reset and SDMMC2 DLL software reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_TAP_SEL_ACK**:
1: The RX delay switched to the tap value specified in RX_TAP_SEL[5:0].

Bit 0 **LOCK**:
1: The DLL is locked.

23.2.9 SYSCFG SDMMC2 control register (SYSCFG_SDMMC2CR)

Address offset: 0x808

Reset value: 0x0000 0000

Reset by system reset and SDMMC2 software reset.

Write access according to SDMMC2 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2_VSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SDMMC2_VSEL**: Voltage selection

Change the value of the SDVSEL2 pin output to control the voltage on pins for external card. Need AFMUX selection.

1: Change the voltage to low-voltage range (1.8 V typ).

0: Change the voltage to high-voltage range (3.3 V typ).

Note: There must be a control circuit out of SDMMC2 to change the voltage on pins.

23.2.10 SYSCFG SDMMC2 address remapping control register (SYSCFG_SDMMC2ARCR)

Address offset: 0x80C

Reset value: 0x0000 0000

Reset by system reset.

Write access according to SDMMC2 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AREN**: Address remapping enable

Enable the address remapping logic in front of the SDMMC2 AHB master. Remapping allows SDMMC2 (only 32bit addressing supported by the master) to access the 4GB of the DDR.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.11 SYSCFG DLYBSD3 control register (SYSCFG_DLYBSD3CR)

Address offset: 0xC00

Reset value: 0x0040 0000

Reset by system reset and SDMMC3 DLL software reset.

Write access according to SDMMC3 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL[5:0]						EN
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **ANTIGLITCH_EN**: Anti-glitch logic enable

1: The anti-glitch logic is enabled. This ensures that there is no glitch on the clock line when the tap value is modified.

Bits 21:17 **BYP_CMD[4:0]**: Delay unit when in DLL bypass mode

00000: Reserved

00001: 1 delay element enabled

00010: 2 delay elements enabled

...

11000: all 24 delay elements enabled

11001: all 24 delay elements enabled

...

11111: all 24 delay elements enabled

Bit 16 **BYP_EN**: DLL bypass mode enable

1: DLL bypass mode is enabled. In bypass mode the DLL is used in pure delay mode and there is no VT compensation. The command applied to every delay unit is set by DLL_BYP_CMD[4:0] and a tap value must be selected with RX_TAP_SEL[5:0]. In bypass mode, the total delay is given by (delay unit) x tap number selected.

0: DLL lock mode is selected.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:1 **RX_TAP_SEL[5:0]**: TAP delay for RX clock selection
000000: 0 tap selected
000001: 1 tap selected
....
000010: 2 taps selected
....
011111: 31 taps selected
100000: 32 taps selected

Bit 0 **EN**:
1: The DLL is enabled when configured in lock mode (BYP_EN = 0).

23.2.12 SYSCFG DLYBSD3 status register (SYSCFG_DLYBSD3SR)

Address offset: 0xC04

Reset value: 0x0000 0002

Reset by system reset and SDMMC3 DLL software reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_TAP_SEL_ACK**:
1: The RX delay has switched to the tap value specified in RX_TAP_SEL[5:0].

Bit 0 **LOCK**:
1: The DLL is locked.

23.2.13 SYSCFG SDMMC3 address remapping control register (SYSCFG_SDMMC3ARCR)

Address offset: 0xC0C

Reset value: 0x0000 0000

Reset by system reset.

Write access according to SDMMC3 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AREN**: Address remapping enable

Enable the address remapping logic in front of the SDMMC3 AHB master. Remapping allows SDMMC3 (only 32bit addressing supported by the master) to access the 4GB of the DDR.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.14 SYSCFG DLYBOS1 control register (SYSCFG_DLYBOS1CR)

Address offset: 0x1000

Reset value: 0x0000 0000

Reset by system reset and OCTOSPI1 DLL software reset.

Write access according to OCTOSPI1 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYP_CMD[4:0]					BYP_EN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TX_TAP_SEL[5:0]						RX_TAP_SEL[5:0]						EN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:17 **BYP_CMD[4:0]**: Delay unit when in DLL bypass mode
00000: Reserved
00001: 1 delay element enabled
00010: 2 delay elements enabled
...
11000: all 24 delay elements enabled
11001: all 24 delay elements enabled
...
11111: all 24 delay elements enabled

Bit 16 **BYP_EN**: DLL bypass mode enable
1: DLL bypass mode is enabled. In bypass mode the DLL is used in pure delay mode and there is no VT compensation. The command applied to every delay unit is set by DLL_BYP_CMD[4:0] and a tap value must be selected with RX_TAP_SEL[5:0]. In bypass mode, the total delay is given by (delay unit) x tap number selected.
0: DLL lock mode is selected.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:7 **TX_TAP_SEL[5:0]**: TAP delay for TX clock selection
000000: 0 tap selected
000001: 1 tap selected
....
000010: 2 taps selected
....
011111: 31 taps selected
100000: 32 taps selected

Bits 6:1 **RX_TAP_SEL[5:0]**: TAP delay for RX clock selection
000000: 0 tap selected
000001: 1 tap selected
....
000010: 2 taps selected
....
011111: 31 taps selected
100000: 32 taps selected

Bit 0 **EN**:
1: The DLL is enabled when configured in lock mode (BYP_EN = 0).

23.2.15 SYSCFG DLYBOS1 status register (SYSCFG_DLYBOS1SR)

Address offset: 0x1004

Reset value: 0x0000 0006

Reset by system reset and OCTOSPI1 DLL software reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TAP_SEL_ACK	RX_TAP_SEL_ACK	LOCK
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TX_TAP_SEL_ACK**:
1: The TX delay has switched to the tap value specified in TX_TAP_SEL[5:0].

Bit 1 **RX_TAP_SEL_ACK**:
1: The RX delay has switched to the tap value specified in RX_TAP_SEL[5:0].

Bit 0 **LOCK**:
1: The DLL is locked.

23.2.16 SYSCFG potential tamper reset control register (SYSCFG_POTTAMPRSTCR)

Address offset: 0x1804

Reset value: 0x0000 0000

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POTTAMPRESETMASK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **POTTAMPRESETMASK**:
This bit can be set by software to mask PKA, SAES, CRYP1/2, and HASH reset in case of potential tamper.
0: PKA, SAES, CRYP1/2, and HASH reset in case of potential tamper
1: PKA, SAES, CRYP1/2, and HASH not reset in case of potential tamper

23.2.17 SYSCFG M33SS control register (SYSCFG_M33SSCR)

Address offset: 0x1C00

Reset value: 0x0000 0000

Reset by system reset.

Write access according to ICACHE_DCACHE RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKNSMPU	LOCKSMPU	LOCKNSVTOR	LOCKSVTAIRCR
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IT_EN[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LOCKSAU**:
Prevent changes to secure SAU memory regions already programmed.

Bit 19 **LOCKNSMPU**:
Prevent changes to non-secure MPU memory regions already programmed.

Bit 18 **LOCKSMPU**:
Prevent changes to programmed secure MPU memory regions.

Bit 17 **LOCKNSVTOR**:
Prevent changes to non-secure vector table base address.

Bit 16 **LOCKSVTAIRCR**:
Prevent changes to:
– secure vector table base address
– secure interrupt priority handling
– BusFault, HardFault, and NMI security target settings in the processor

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IT_EN[5:0]**: Floating point unit interrupt enable
FPU_IT_EN[5]: Input abnormal interrupt enable
FPU_IT_EN[4]: Inexact interrupt enable
FPU_IT_EN[3]: Underflow interrupt enable
FPU_IT_EN[2]: Overflow interrupt enable
FPU_IT_EN[1]: Divide-by-zero interrupt enable
FPU_IT_EN[0]: Invalid operation Interrupt enable

23.2.18 SYSCFG ICN QOS priority control register 1 (SYSCFG_ICNQPCR1)

Address offset: 0x2000

Reset value: 0xCCCB BB3E

Reset by system reset.
Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDMMC3_QOS[3:0]				SDMMC2_QOS[3:0]				SDMMC1_QOS[3:0]				HPDMA3_QOS[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPDMA2_QOS[3:0]				HPDMA1_QOS[3:0]				TRACE_QOS[3:0]				A35_QOS[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SDMMC3_QOS[3:0]**: SDMMC3 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 27:24 **SDMMC2_QOS[3:0]**: SDMMC2 read/write channes QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 23:20 **SDMMC1_QOS[3:0]**: SDMMC1 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 19:16 **HPDMA3_QOS[3:0]**: HPDMA3 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 15:12 **HPDMA2_QOS[3:0]**: HPDMA2 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 11:8 **HPDMA1_QOS[3:0]**: HPDMA1 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 7:4 **TRACE_QOS[3:0]**: TRACE read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 3:0 **A35_QOS[3:0]**: Cortex-A35 read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

23.2.19 SYSCFG ICN QOS priority control register 2 (SYSCFG_ICNQPCR2)

Address offset: 0x2004

Reset value: 0x0009 80AA

Reset by system reset.
Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDC_QOS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMIPP_QOS[3:0]				Res.	Res.	Res.	Res.	OTG_QOS[3:0]				USBH_QOS[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **LTDC_QOS[3:0]**: LTDC read/write channels QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 15:12 **DCMIPP_QOS[3:0]**: DCMIPP write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **OTG_QOS[3:0]**: OTG read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

Bits 3:0 **USBH_QOS[3:0]**: USBH(OHCI and EHCI) read/write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

23.2.20 SYSCFG ICN early write response control register (SYSCFG_ICNEWRCR)

Address offset: 0x2008

Reset value: 0x0000 0000

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTG_EARLY_WR_RSP_ENABLE	Res.	SDMMC3_EARLY_WR_RSP_ENABLE	SDMMC2_EARLY_WR_RSP_ENABLE	SDMMC1_EARLY_WR_RSP_ENABLE	USBH_EARLY_WR_RSP_ENABLE
										rw		rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OTG_EARLY_WR_RSP_ENABLE**: OTG early write response enable

0: Early write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.

1: Early write response enabled. AHB-Lite write data beats receive an automatic OKAY response from the AHB-to-AXI bridge irrespective of the B-channel AXI response.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SDMMC3_EARLY_WR_RSP_ENABLE**: SDMMC3 early write response enable

0: Early write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.

1: Early write response enabled. AHB-Lite write data beats receive an automatic OKAY response from the AHB-to-AXI bridge irrespective of the B-channel AXI response.

Bit 2 **SDMMC2_EARLY_WR_RSP_ENABLE**: SDMMC2 early write response enable

0: Early write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.

1: Early write response enabled. AHB-Lite write data beats receive an automatic OKAY response from the AHB-to-AXI bridge irrespective of the B-channel AXI response.

Bit 1 **SDMMC1_EARLY_WR_RSP_ENABLE**: SDMMC1 early write response enable

0: Early write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.

1: Early write response enabled. AHB-Lite write data beats receive an automatic OKAY response from the AHB-to-AXI bridge irrespective of the B-channel AXI response.

Bit 0 **USBH_EARLY_WR_RSP_ENABLE**: USBH (OHCI and EHCI) early write response enable

0: Early write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.

1: Early write response enabled. AHB-Lite write data beats receive an automatic OKAY response from the AHB-to-AXI bridge irrespective of the B-channel AXI response.

23.2.21 SYSCFG ICN end-to-end bandwidth regulation control register (SYSCFG_ICNE2EBWRCR)

Address offset: 0x2014

Reset value: 0x0000 0052

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CFG_HPDMA_BW	CFG_CPU1_BW	Res.	CFG_ALL_TGT[2:0]			Res.	CFG_MODEL[1:0]		CFG_ENABLE
						rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CFG_HPDMA_BW**:
Triple the bandwidth (BW) allowed to HPDMA1/2/3 by the regulation.
0: Normal bandwidth allocated to HPDMA1/2/3
1: Increased bandwidth allocated to HPDMA1/2/3
Note: This additional bandwidth is preempted from other peripherals, that may thus face real-time issues. Keep default setting unless ST recommendation.

Bit 8 **CFG_CPU1_BW**:
Triple the bandwidth allowed to the CPU1 by the regulation.
0: Normal bandwidth allocated to HPDMA1/2/3
1: Increased bandwidth allocated to HPDMA1/2/3
Note: This additional bandwidth is preempted from other peripherals, that may thus face real-time issues. Keep default setting unless ST recommendation.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CFG_ALL_TGT[2:0]**: Amount of empty CAM slots in DDR-CTRL that the regulation targets
000: 9 empty slots targeted
001: 11 empty slots targeted
010: 13 empty slots targeted
011: 15 empty slots targeted
100: 17 empty slots targeted
101: 20 empty slots targeted (default)
110: 24 empty slots targeted
111: 28 empty slots targeted

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **CFG_MODEL[1:0]**:
Defines the reactivity of the regulation feedback.
00: Slow reaction (risk of congestion if the regulation reacts too slow)
11: Fast reaction (risk of oscillation if the regulation reacts too fast)

Bit 0 **CFG_ENABLE**:

Activates the end-to-end bandwidth regulation.

0: Regulation disabled. Peripheral bus masters access without constrain into the STNoC AXI network-on-chip.

1: Regulation enabled. The regulation favors the high QoS, and limits some bandwidths.

23.2.22 SYSCFG safe reset control register (SYSCFG_SAFERSTCR)

Address offset: 0x2018

Reset value: 0x0000 0000

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG_ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CFG_ENABLE**:

Activates the safe reset connector.

0: Safe reset connector disabled

1: Safe reset connector enabled

23.2.23 SYSCFG ICN ETH bandwidth limiter control register (SYSCFG_ICNETHBWLCR)

Address offset: 0x2020

Reset value: 0x0000 1002

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **MAXABW[7:0]**: Maximum allowed bandwidth

Configures the maximum allowed bandwidth for each ETH (read and write traffic) in proportion of the link throughput in one direction, 2.4 GB/s with a default clock configuration:

0: Reserved
1: Bandwidth limited at 1/128 of 2.4 GB/s.
2: Bandwidth limited at 2/128 of 2.4 GB/s.
255: Bandwidth limited at 255/128 of 2.4 GB/s.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BWLEN**: Bandwidth limiter enable

Activates the ETHs bandwidth limiter.

0: Bandwidth limiter disabled
1: Bandwidth limiter enabled

Bit 0 Reserved, must be kept at reset value.

23.2.24 SYSCFG ICN CPU1 bandwidth limiter control register (SYSCFG_ICNCPU1BWLCR)

Address offset: 0x2028

Reset value: 0x0000 1802

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **MAXABW[7:0]**: Maximum allowed bandwidth

Configures the maximum allowed bandwidth for the CPU1 (read and write traffic) in proportion of the link throughput in one direction, 9.6 GB/s with a default clock configuration:

0: Reserved
1: Bandwidth limited at 1/128 of 9.6 GB/s.
2: Bandwidth limited at 2/128 of 9.6 GB/s.
255: Bandwidth limited at 255/128 of 9.6 GB/s.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BWLEN**: Bandwidth limiter enable

Activates the CPU1 bandwidth limiter.

0: Bandwidth limiter disabled
1: Bandwidth limiter enabled

Bit 0 Reserved, must be kept at reset value.

23.2.25 SYSCFG ICN LTDC bandwidth limiter control register (SYSCFG_ICNLTDCBWLCR)

Address offset: 0x202C

Reset value: 0x0000 1E02

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWL EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **MAXABW[7:0]**: Maximum allowed bandwidth

Configures the maximum allowed bandwidth for the LTDC (read and write traffic) in proportion of the link throughput in one direction, 3.2 GB/s with a default clock configuration:

0: Reserved
1: Bandwidth limited at 1/128 of 3.2 GB/s.
2: Bandwidth limited at 2/128 of 3.2 GB/s.
255: Bandwidth limited at 255/128 of 3.2 GB/s.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BWLEN**: Bandwidth limiter enable

Activates the LTDC bandwidth limiter.
0: Bandwidth limiter disabled
1: Bandwidth limiter enabled

Bit 0 Reserved, must be kept at reset value.

23.2.26 SYSCFG ICN DCMIPP bandwidth limiter control register (SYSCFG_ICNDCMIPPBWLCR)

Address offset: 0x2030

Reset value: 0x0000 2402

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWL EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **MAXABW[7:0]**: Maximum allowed bandwidth

Configures the maximum allowed bandwidth for the DCMIPP (only write traffic) in proportion of the link throughput in one direction, 3.2 GB/s with a default clock configuration:

0: Reserved

1: Bandwidth limited at 1/128 of 3.2GB/s.

2: Bandwidth limited at 2/128 of 3.2GB/s.

255: Bandwidth limited at 255/128 of 3.2GB/s.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BWLEN**: Bandwidth limiter enable

Activates the DCMIPP bandwidth limiter.

0: Bandwidth limiter disabled

1: Bandwidth limiter enabled

Bit 0 Reserved, must be kept at reset value.

23.2.27 SYSCFG ICN HPDMA bandwidth limiter control register (SYSCFG_ICNHPDMABWLCR)

Address offset: 0x2038

Reset value: 0x0000 4802

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWL EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **MAXABW[7:0]**: Maximum allowed bandwidth

Configures the maximum allowed bandwidth for the HPDMA (only write traffic) in proportion of the link throughput in one direction, 3.2 GB/s with a default clock configuration:

0: Reserved

1: Bandwidth limited at 1/128 of 3.2GB/s.

2: Bandwidth limited at 2/128 of 3.2GB/s.

255: Bandwidth limited at 255/128 of 3.2GB/s.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **BWLEN**: Bandwidth limiter enable

Activates the HPDMA bandwidth limiter.

0: Bandwidth limiter disabled

1: Bandwidth limiter enabled

Bit 0 Reserved, must be kept at reset value.

23.2.28 SYSCFG HPDMA address remapping control register (SYSCFG_HPDMAARCR)

Address offset: 0x2050

Reset value: 0x0000 0000

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA3AREN	HPDMA2AREN	HPDMA1AREN
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **HPDMA3AREN**: Address remapping enable

Enable the address remapping logic in front of the HPDMA3 AXI master. Remapping allows HPDMA3 AXI master (only 32bit addressing supported by the master) to access the 4GB of the DDR. The AHB master is not impacted by the remapping logic.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

Bit 1 **HPDMA2AREN**: Address remapping enable

Enable the address remapping logic in front of the HPDMA2 AXI master. Remapping allows HPDMA2 AXI master (only 32bit addressing supported by the master) to access the 4GB of the DDR. The AHB master is not impacted by the remapping logic.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

Bit 0 **HPDMA1AREN**: Address remapping enable

Enable the address remapping logic in front of the HPDMA1 AXI master. Remapping allows HPDMA1 AXI master (only 32bit addressing supported by the master) to access the 4GB of the DDR. The AHB master is not impacted by the remapping logic.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.29 SYSCFG USB2PHY1 control register (SYSCFG_USB2PHY1CR)

Address offset: 0x2400

Reset value: 0x0000 0015

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1SEL[2:0]			Res.	USB2PHY1CMN	AUTORSMENB1	RETENABLEN1
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **USB2PHY1SEL[2:0]**: PHY reference clock speed setting
000: 19.2 MHz
001: 20 MHz (default)
010: 24 MHz
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY1CMN**:
Controls power down of analog blocks during Suspend and Sleep.
0: PHY clocks are kept active during Suspend and Sleep (with the exception of PHYCLK). The PLL is on. Requires to keep the PHY reference clock on.
1: PHY clocks are off during Suspend and Sleep. PLL is off (default).

Bit 1 **AUTORSMENB1**: Auto-resume mode enable
Enables auto-resume logic in femtoPHY. The PHY automatically responds to a remote wake-up without initial involvement of the Host controller.
0: Auto-resume disabled (default
1: A pulse on this bit prior to enabling the retention mode, enables the auto-resume mode.

Bit 0 **RETENABLEN1**: Retention mode enable (active low)
0: Retention mode enabled
1: Retention mode disabled (default)

23.2.30 SYSCFG USB2PHY1 battery charging control register (SYSCFG_USB2PHY1BCCR)

Address offset: 0x2404

Reset value: 0x0000 0000

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H_FORCEBCMODE	H_VDMSRCEN	H_CDPDETEN	H_CDPEN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **H_FORCEBCMODE**:

Forces USB2PHY1 in non-driving suspend mode. This is the recommended mode to perform battery charging procedure.

0: Normal mode. SUSPENDM0 and OPMODE0 inputs of USB2PHY1 are managed by the USB2 Host controller (default).

1: OPMODE0[1:0] is forced to non-driving, and SUSPENDM0 is forced to active.

Bit 2 **H_VDMSRCEN**: Voltage source on DM for CDP port enable

0: Voltage source disabled (default)

1: Voltage source enabled

Bit 1 **H_CDPDETEN**: Voltage detector on DP for CDP port enable

0: voltage detector on DP disabled (default)

1: voltage detector on DP enabled

Bit 0 **H_CDPEN**: CDP behavior for Host port enable

0: CDP behavior disabled (Host port behaves as an SDP) (default)

1: CDP behavior enabled

23.2.31 SYSCFG USB2PHY1 battery charging status register (SYSCFG_USB2PHY1BCSR)

Address offset: 0x2408

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSVMINUS	FSVPLUS	CHGDET
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSVMINUS**: Voltage level on DM (also output of the comparison with VLGC threshold, as defined in BC v1.2 standard)
0: VDM < VLGC
1: VDM > VLGC

Bit 1 **FSVPLUS**: Voltage level on DP (also output of the comparison with VLGC threshold, as defined in BC v1.2 standard)
0: VDP < VLGC
1: VDP > VLGC

Bit 0 **CHGDET**: Comparison of voltage on DP with VDAT_REF threshold as defined in BC v1.2 standard
0: VDP < VDAT_REF
1: VDP > VDAT_REF

23.2.32 SYSCFG USB2PHY1 trimming 1 control register (SYSCFG_USB2PHY1TRIM1CR)

Address offset: 0x240C

Reset value: 0x299E D6F0

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	USB2PHY1TXRESTUNE[1:0]		USB2PHY1TXRISETUNE[1:0]		USB2PHY1TXVREFTUNE[3:0]				USB2PHY1TXFSLSTUNE[3:0]				USB2PHY1TXHSXVTUNE[1:0]		USB2PHY1OTGTUNE[2]
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB2PHY1OTGTUNE[1:0]		USB2PHY1VDATREFTUNE[1:0]		USB2PHY1SQRXTUNE[2:0]			USB2PHY1COMPDISTUNE[2:0]			USB2PHY1PLLPTUNE[3:0]				USB2PHY1PLLITUNE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **USB2PHY1TXRESTUNE[1:0]**: USB source impedance adjustment

In some applications, there is significant series resistance on DP and DM paths between the transceiver and the cable. This bitfield adjusts the driver source impedance to compensate for added series resistance on the USB. Any setting other than the default one can result in source impedance variation across process, voltage and temperature that does not meet USB2.0 specifications limits.

11: Source impedance decreased by approximately 5.6 ?

10: Source impedance decreased by approximately 3.6 ?

01: Nominal

00: Source impedance increased by approximately 1.3 ?

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 28:27 **USB2PHY1TXRISETUNE[1:0]**: High-speed transmitter rise/fall time adjustment

This bitfield is used to adjust rise/fall times of the high-speed waveform.

11: -8.1%

10: -7.2%

01: Nominal

00: +5.4%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 26:23 **USB2PHY1TXVREFTUNE[3:0]**: HS DC voltage level adjustment

This bitfield is used to adjust the high-speed (HS) DC level voltage. When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE) may also request a change to the HS disconnect detection threshold (COMPDISTUNE).

1111: +24%
1110: +22%
1101: +20%
1100: +18%
1011: +16%
1010: +14%
1001: +12%
1000: +10%
0111: +8%
0110: +6%
0101: +4%
0100: +2%
0011: Nominal
0010: -2%
0001: -4%
0000: -6%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 22:19 **USB2PHY1TXFSLSTUNE[3:0]**: Full-speed (FS)/low-speed (LS) source impedance adjustment

This bit field is used to adjust FS and LS single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage and temperature.

1111: -10.7%
0111: -5.6%
0011: Nominal
0001: +6.5%
0000: +14%
Other: reserved

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 18:17 **USB2PHY1TXHSXVTUNE[1:0]**: Transmitter high-speed crossover adjustment

This bitfield is used to adjust the voltage at which DP and DM signals cross while transmitting in high-speed mode.

11: Nominal
10: +15 mV
01: -15 mV
00: Reserved

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 16:14 **USB2PHY1OTGTUNE[2:0]**: V_{BUS} valid threshold adjustment

This bitfield is used to adjust the voltage level for the V_{BUS} valid threshold.

111: +12%
110: +9%
101: +6%
100: +3%
011: Nominal
010: -3%
001: -6%
000: -9%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 13:12 **USB2PHY1VDATREFTUNE[1:0]**: Data detect voltage adjustment

This bitfield is used to adjust the threshold voltage level (Vdat_ref) used to detect data during charger type detection.

11: -20%
10: -10%
01: Nominal
00: +10%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 11:9 **USB2PHY1SQRXTUNE[2:0]**: Squelch threshold adjustment

This bitfield is used to adjust the voltage level for the threshold used to detect valid HS data.

111: -20%
110: -15%
101: -10%
100: -5%
011: Nominal
010: +5%
001: +10%
000: +15%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 8:6 **USB2PHY1COMPDISTUNE[2:0]**: Disconnect threshold adjustment

This bitfield adjusts the voltage level for the threshold used to detect a disconnect event at the Host. When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE) may also request a change to the HS disconnect detection threshold (COMPDISTUNE).

111: +12%
110: +10%
101: +7%
100: +3%
011: Nominal
010: -4%
001: -6%
000: -9%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 5:2 **USB2PHY1PLLPTUNE[3:0]**: PLL proportional path tune

1111: 5.5x
1110: 5x
1101: 4.5x
1100: 4x
1011: 4.5x
1010: 4x
1001: 3.5x
1000: 3x
0111: 3.5x
0110: 3x
0101: 2.5x
0100: 2x
0011: 2.5x
0010: 2x
0001: 1.5x
0000: 1x

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 1:0 **USB2PHY1PLLITUNE[1:0]**: PLL integral path tune

11: 2.0x
10: 1.5x
01: 1.0x
00: 0.5x

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

23.2.33 SYSCFG USB2PHY1 trimming 2 control register (SYSCFG_USB2PHY1TRIM2CR)

Address offset: 0x2410

Reset value: 0x0000 0000

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1TXPREEMPPULSETUNE	USB2PHY1TXPREEMPAMPTUNE[1:0]	
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY1TXPREEMPPULSETUNE**: HS transmitter preemphasis duration control

This bit controls the duration for which the HS preemphasis current is sourced onto DM or DP. The HS transmitter preemphasis duration is defined in terms of unit amounts. One unit of preemphasis duration is approximately 580 ps, and is defined as 1x preemphasis duration. This bit is valid only if TXPREEMPAMPTUNE[0] or TXPREEMPAMPTUNE[1] is set to 1.

1: 1x, short pre-emphasis current duration

0: 2x, long pre-emphasis current duration

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 1:0 **USB2PHY1TXPREEMPAMPTUNE[1:0]**: HS transmitter preemphasis current control

This bitfield controls the amount of current sourced to DP and DM after a J-to-K or K-to-J transition. The HS transmitter preemphasis current is defined in terms of unit amounts. One unit amount is approximately 600 ?A, and is defined as 1x pre-emphasis current.

11: HS transmitter preemphasis circuit sources 3x preemphasis current

10: HS transmitter preemphasis circuit sources 2x preemphasis current

01: HS transmitter preemphasis circuit sources 1x preemphasis current

00: HS transmitter preemphasis disabled

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

23.2.34 SYSCFG USBH control register (SYSCFG_USBHCR)

Address offset: 0x2420

Reset value: 0x0000 0000

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBH_VBUSEN_POLARITY	USBH_OVRCUR_POLARITY
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **USBH_VBUSEN_POLARITY**: Polarity of USBH_VBUSEN alternate function

0: USBH_VBUSEN active high (a 1 logic level means VBUS must be enabled) (default)

1: USBH_VBUSEN active low (a 0 logic level means VBUS must be enabled)

Bit 0 **USBH_OVRCUR_POLARITY**: Polarity of USBH_OVRCUR alternate function

0: USBH_OVRCUR active high (a 1 logic level means there is an over-current situation) (default)

1: USBH_OVRCUR active low (a 0 logic level means there is an over-current situation)

23.2.35 SYSCFG USBH address remapping control register (SYSCFG_USBHARCR)

Address offset: 0x2424

Reset value: 0x0000 0000

Reset by system reset.

Write access according to USBH RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AREN**: Address remapping enable

Enable the address remapping logic in front of the USBH AHB masters (EHCI and OHCI). Remapping allows USBH (only 32bit addressing supported by the master) to access the 4GB of the DDR.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.36 SYSCFG USB2PHY2 control register (SYSCFG_USB2PHY2CR)

Address offset: 0x2800

Reset value: 0x0000 0015

Reset by system reset.

Write access according to OTG RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2SEL[2:0]			Res.	USB2PHY2CMN	AUTORSMENB2	RETENABLEN2
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **USB2PHY2SEL[2:0]**: PHY reference clock speed setting
000: 19.2 MHz
001: 20 MHz (default)
010: 24 MHz
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY2CMN**: Control of power down of analog blocks during Suspend and Sleep
0: PHY clocks are kept active during Suspend and Sleep (with the exception of PHYCLK). The PLL is on. Requires to keep the PHY reference clock on.
1: PHY clocks are off during Suspend and Sleep. PLL is off (default)

Bit 1 **AUTORSMENB2**: Auto-resume mode enable
This bit enables auto-resume logic in femtoPHY. The PHY automatically responds to a remote wake-up without initial involvement of the Host controller.
0: Auto-resume is disabled (default).
1: A pulse on this bit prior to enabling the retention mode, enables the auto-resume mode.

Bit 0 **RETENABLEN2**: Retention mode enable (active low)
0: Retention mode enabled
1: Retention mode disabled (default)

23.2.37 SYSCFG USB2PHY2 trimming 1 control register (SYSCFG_USB2PHY2TRIM1CR)

Address offset: 0x2808

Reset value: 0x299E D6F0

Reset by system reset.

Write access according to OTG RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	USB2PHY2TXRESTUNE[1:0]		USB2PHY2TXRISETUNE[1:0]		USB2PHY2TXVREFTUNE[3:0]				USB2PHY2TXFSLSTUNE[3:0]				USB2PHY2TXHSXVTUNE[1:0]		USB2PHY2OTGTUNE[2]
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB2PHY2OTGTUNE[1:0]		USB2PHY2VDATREFTUNE[1:0]		USB2PHY2SQRXTUNE[2:0]			USB2PHY2COMPDISTUNE[2:0]			USB2PHY2PLLPTUNE[3:0]				USB2PHY2PLLITUNE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **USB2PHY2TXRESTUNE[1:0]**: USB source impedance adjustment

In some applications there is significant series resistance on DP and DM paths between the transceiver and the cable. This bitfield is used to adjust the driver source impedance to compensate for added series resistance on the USB. Any setting other than the default one can result in source impedance variation across process, voltage and temperature that does not meet USB2.0 specifications limits.

11: Source impedance decreased by approximately 5.6 ?

10: Source impedance decreased by approximately 3.6 ?

01: Nominal

00: Source impedance increased by approximately 1.3 ?

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 28:27 **USB2PHY2TXRISETUNE[1:0]**: HS transmitter rise/fall time adjustment

This bitfield is used to adjust the rise/fall times of the high-speed waveform.

11: -8.1%

10: -7.2%

01: Nominal

00: +5.4%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 26:23 **USB2PHY2TXVREFTUNE[3:0]**: HS DC voltage level adjustment

This bitfield is used to adjust the HS DC level voltage. When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE) may also request a change to the HS disconnect detection threshold (COMPDISTUNE).

1111: +24%

1110: +22%

1101: +20%

1100: +18%

1011: +16%

1010: +14%

1001: +12%

1000: +10%

0111: +8%

0110: +6%

0101: +4%

0100: +2%

0011: Nominal

0010: -2%

0001: -4%

0000: -6%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 22:19 **USB2PHY2TXFSLSTUNE[3:0]**: Full-speed (FS)/low-speed (LS) source impedance adjustment

This bitfield is used to adjust the FS and LS single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage and temperature.

1111: -10.7%

0111: -5.6%

0011: Nominal

0001: +6.5%

0000: +14%

Other: reserved

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 18:17 **USB2PHY2TXHSXVTUNE[1:0]**: Transmitter HS crossover adjustment

This bitfield is used to adjust the voltage at which DP and DM signals cross while transmitting in HS mode.

11: Nominal

10: +15 mV

01: -15 mV

00: Reserved

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 16:14 **USB2PHY2OTGTUNE[2:0]**: VBUS valid threshold adjustment

This bitfield is used to adjust the voltage level for the VBUS valid threshold.

111: +12%

110: +9%

101: +6%

100: +3%

011: Nominal

010: -3%

001: -6%

000: -9%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 13:12 **USB2PHY2VDATREFTUNE[1:0]**: Data detect voltage adjustment

This bitfield is used to adjust the threshold voltage level (Vdat_ref) used to detect data during charger type detection.

11: -20%

10: -10%

01: Nominal

00: +10%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 11:9 **USB2PHY2SQRXTUNE[2:0]**: Squelch threshold adjustment

This bitfield is used to adjust the voltage level for the threshold used to detect valid HS data.

111: -20%
110: -15%
101: -10%
100: -5%
011: Nominal
010: +5%
001: +10%
000: +15%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 8:6 **USB2PHY2COMPDISTUNE[2:0]**: Disconnect threshold adjustment

This bitfield is used to adjust the voltage level for the threshold used to detect a disconnect event at the Host. When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE) may also necessitate a change to the HS disconnect detection threshold (COMPDISTUNE).

111: +12%
110: +10%
101: +7%
100: +3%
011: Nominal
010: -4%
001: -6%
000: -9%

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 5:2 **USB2PHY2PLLPTUNE[3:0]**: PLL proportional path tune

1111: 5.5x
1110: 5x
1101: 4.5x
1100: 4x
1011: 4.5x
1010: 4x
1001: 3.5x
1000: 3x
0111: 3.5x
0110: 3x
0101: 2.5x
0100: 2x
0011: 2.5x
0010: 2x
0001: 1.5x
0000: 1x

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 1:0 **USB2PHY2PLLITUNE[1:0]**: PLL integral path tune

11: 2.0x

10: 1.5x

01: 1.0x

00: 0.5x

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

23.2.38 SYSCFG USB2PHY2 trimming 2 control register (SYSCFG_USB2PHY2TRIM2CR)

Address offset: 0x280C

Reset value: 0x0000 0000

Reset by system reset.

Write access according to OTG RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2TXPREEMPPULSETUNE	USB2PHY2TXPREEMPAMPTUNE[1:0]	
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **USB2PHY2TXPREEMPPULSETUNE**: HS transmitter preemphasis duration control

This bit controls the duration for which the HS preemphasis current is sourced onto DM or DP. The HS transmitter preemphasis duration is defined in terms of unit amounts. One unit of preemphasis duration is approximately 580 ps and is defined as 1x preemphasis duration. This bit is valid only if TXPREEMPAMPTUNE[0] or TXPREEMPAMPTUNE[1] is set to 1.

1: 1x, short preemphasis current duration

0: 2x, long preemphasis current duration

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

Bits 1:0 **USB2PHY2TXPREEMPAMPTUNE[1:0]**: HS transmitter preemphasis current control

This bitfield controls the amount of current sourced to DP and DM after a J-to-K or K-to-J transition. The HS transmitter preemphasis current is defined in terms of unit amounts. One unit amount is approximately 600 ?A and is defined as 1x pre-emphasis current.

11: HS transmitter preemphasis circuit sources 3x preemphasis current

10: HS transmitter preemphasis circuit sources 2x preemphasis current

01: HS transmitter preemphasis circuit sources 1x preemphasis current

00: HS transmitter preemphasis is disabled

Note: A strapping option must be set prior to USB2PHY reset, and must remain static during normal operation.

23.2.39 SYSCFG OTG address remapping control register (SYSCFG_OTGARCR)

Address offset: 0x2824

Reset value: 0x0000 0000

Reset by system reset.

Write access according to OTG RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **AREN**: Address remapping enable

Enable the address remapping logic in front of the OTG AHB master. Remapping allows OTG (only 32bit addressing supported by the master) to access the 4GB of the DDR.

1: Address remapping is enabled. The 4GB address space accessible by the master starts at address 0x80000000.

0: Address remapping is not enabled. The 4GB address space accessible by the master starts at address 0x0.

23.2.40 SYSCFG Ethernet1 control register (SYSCFG_ETH1CR)

Address offset: 0x3000

Reset value: 0x0000 0000

Reset by system reset.

Write access according to ETH1 RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1_SEL[2:0]			Res.	Res.	ETH1_CLK_SEL	ETH1_REF_CLK_SEL
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **ETH1_SEL[2:0]**: Ethernet 1 PHY interface selection
000: MII
001: RGMII
100: RMII
Others: Reserved
Note: Configuration must be done while ETH1 is under reset and before enabling ETH1 clocks.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ETH1_CLK_SEL**: RGMII 125 MHz clock selection
0: External clock (ETH1_CLK125) is used. Need to program AFMUX.
1: Internal clock from the RCC is used.

Bit 0 **ETH1_REF_CLK_SEL**: Ethernet reference/RX clock selection
0: External clock is used. This is RX clock for an RGMII or MII PHY, or REF clock for an RMII PHY. Need to program AFMUX.
1: Internal clock from the RCC is used. To be used when we generate the RMII 50 MHz (ETH1_CLK pin) to RMII PHY.

23.2.41 SYSCFG Ethernet1 status register (SYSCFG_ETH1SR)

Address offset: 0x3010

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1_PWR_DOWN_ACK	ETH1_TX_CLK_GATING_CTRL
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ETH1_PWR_DOWN_ACK**: Asserted when the power-down sequence start has been acknowledged

Bit 0 **ETH1_TX_CLK_GATING_CTRL**: Asserted when the MAC has entered Tx LPI mode

23.2.42 SYSCFG Ethernet2 configuration register (SYSCFG_ETH2CR)

Address offset: 0x3400

Reset value: 0x0000 0000

Reset by system reset.

Write access according to ETH2 RIFSC configuration. Debug domain can also access if the

feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH2_SEL[2:0]			Res.	Res.	ETH2_CLK_SEL	ETH2_REF_CLK_SEL
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **ETH2_SEL[2:0]**: Ethernet 2 PHY interface selection
000: MII
001: RGMII
100: RMII
Others: Reserved

Note: Configuration must be done while the ETH2 is under reset and before enabling the ETH2 clocks

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ETH2_CLK_SEL**: RGMII 125MHz clock selection
0: External clock (ETH2_CLK125) is used. Need to program AFMUX.
1: Internal clock from the RCC is used.

Bit 0 **ETH2_REF_CLK_SEL**: Reference/RX clock selection
0: External clock is used. This is RX clock for an RGMII or MII PHY, or REF clock for an RMII PHY. Need to program AFMUX.
1: Internal clock from the RCC is used. To be used when we generate the RMII 50 MHz (ETH2_CLK pin) to RMII PHY.

23.2.43 SYSCFG Ethernet2 status register (SYSCFG_ETH2SR)

Address offset: 0x3410

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH2_PWR_DOWN_ACK	ETH2_TX_CLK_GATING_CTRL
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ETH2_PWR_DOWN_ACK**: Asserted when the power-down sequence start has been acknowledged

Bit 0 **ETH2_TX_CLK_GATING_CTRL**: Asserted when the MAC has entered Tx LPI mode

23.2.44 SYSCFG VDDIO3 compensation cell control register (SYSCFG_VDDIO3CCCR)

Address offset: 0x4000

Reset value: 0x0000 0078

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC[3:0]				RANSRC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **LPMDIS**: V_{DDIO3} I/Os Compensation cell low power mode disable.

0: IO compensation is disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

1: IO compensation is not disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

Bit 9 **CS**: I/Os supplied by V_{DDIO3} code selection

This bit selects the code to be applied for the compensation cell of I/Os supplied by V_{DDIO3}.

0: V_{DDIO3} I/O code from the cell (available in SYSCFG_VDDIO3CCSR)

1: V_{DDIO3} I/O code from RANSRC[3:0] and RAPSRC[3:0] register values

Note: CS = 0 is not taken into account until READY = 1. This means that, whenever CS, RANSRC[3:0] and RAPSRC[3:0] are used for I/O compensation when the compensation cell is not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: I/Os supplied by V_{DDIO3} compensation cell enable

This bit enables the compensation cell of the I/Os supplied by V_{DDIO3}.

0: V_{DDIO3} I/Os compensation cell disabled

1: V_{DDIO3} I/Os compensation cell enabled

Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. Similarly, the HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: PMOS compensation code of the I/Os supplied by V_{DDIO3}

These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

Bits 3:0 **RANSRC[3:0]**: NMOS compensation code of the I/Os supplied by V_{DDIO3}

These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

23.2.45 SYSCFG VDDIO3 compensation cell status register (SYSCFG_VDDIO3CCSR)

Address offset: 0x4004

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: V_{DDIO3} I/O compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DDIO3}

0: V_{DDIO3} I/O compensation cell not ready

1: V_{DDIO3} I/O compensation cell ready

Note: The HSI clock is required for the compensation cell to work properly. The compensation cell READY bit is not set if the HSI clock is not enabled.

Bits 7:4 **APSRC[3:0]**: PMOS compensation value of the I/Os supplied by V_{DDIO3}

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO3CCCR is reset, and READY is set.

Bits 3:0 **ANSRC[3:0]**: NMOS compensation value of the I/Os supplied by V_{DDIO3}

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO3CCCR is reset, and READY is set.

23.2.46 SYSCFG VDD compensation cell control register (SYSCFG_VDDCCCR)

Address offset: 0x4010

Reset value: 0x0000 0078

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC[3:0]				RANSRC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **LPMDIS**: V_{DD} I/Os Compensation cell low power mode disable.

0: IO compensation is disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

1: IO compensation is not disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

Bit 9 **CS**: V_{DD} I/O code selection

This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DD}.

0: V_{DD} I/O code from the cell (available in SYSCFG_VDDCCSR)

1: V_{DD} I/O code from RANSRC[3:0] and RAPSRC[3:0] values

Note: CS = 0 is not taken into account until READY = 1. This means that whenever CS, RANSRC[3:0] and RAPSRC[3:0] are used for IO compensation when the compensation cell is not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: V_{DD} I/O compensation cell enable

This bit enables the compensation cell of the I/Os supplied by V_{DD}.

0: V_{DD} I/Os compensation cell disabled

1: V_{DD} I/Os compensation cell enabled

Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. Similarly, the HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: PMOS compensation code of the I/Os supplied by V_{DD}

These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

Bits 3:0 **RANSRC[3:0]**: NMOS compensation code of the I/Os supplied by V_{DD}

These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

23.2.47 SYSCFG VDD compensation cell status register (SYSCFG_VDDCCSR)

Address offset: 0x4014

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: V_{DD} I/O compensation cell ready flag
This bit provides the compensation cell status of the I/Os supplied by V_{DD}.
0: V_{DD} I/O compensation cell not ready
1: V_{DD} I/O compensation cell ready
Note: The HSI clock is required for the compensation cell to work properly. The compensation cell READY bit is not set if the HSI clock is not enabled.

Bits 7:4 **APSRC[3:0]**: PMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDCCCR is reset, and the READY bit is set.

Bits 3:0 **ANSRC[3:0]**: NMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDCCCR is reset, and the READY bit is set.

23.2.48 SYSCFG VDDIO2 compensation cell control register (SYSCFG_VDDIO2CCCR)

Address offset: 0x4018

Reset value: 0x0000 0078

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC[3:0]				RANSRC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **LPMDIS**: V_{DDIO2} I/Os Compensation cell low power mode disable.
0: IO compensation is disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.
1: IO compensation is not disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

Bit 9 **CS**: V_{DDIO2} I/O code selection
This bit selects the code to be applied for the compensation cell of I/Os supplied by V_{DDIO2}.
0: V_{DDIO2} I/O code from the cell (available in the SYSCFG_VDDIO2CCSR)
1: V_{DDIO2} I/O code from RANSRC[3:0] and RAPSRC[3:0] register values
Note: CS = 0 is not taken into account until READY = 1. This means that whenever CS, RANSRC[3:0] and RAPSRC[3:0] are used for I/O compensation when the compensation cell is not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: V_{DDIO2} I/O compensation cell enable
This bit enables the compensation cell of the I/Os supplied by V_{DDIO2}.
0: V_{DDIO2} I/O compensation cell disabled
1: V_{DDIO2} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. Similarly, the HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: PMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

Bits 3:0 **RANSRC[3:0]**: NMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

23.2.49 SYSCFG VDDIO2 compensation cell status register (SYSCFG_VDDIO2CCSR)

Address offset: 0x401C

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: V_{DDIO2} I/O compensation cell ready flag
This bit provides the compensation cell status of the I/Os supplied by V_{DDIO2}.
0: V_{DDIO2} I/O compensation cell not ready
1: V_{DDIO2} I/O compensation cell ready
Note: The HSI clock is required for the compensation cell to work properly. The compensation cell READY bit is not set if the HSI clock is not enabled.

Bits 7:4 **APSRC[3:0]**: PMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO2CCCR is reset, and the READY bit is set.

Bits 3:0 **ANSRC[3:0]**: NMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO2CCCR is reset, and the READY bit is set.

23.2.50 SYSCFG VDDIO1 compensation cell control register (SYSCFG_VDDIO1CCCR)

Address offset: 0x4020

Reset value: 0x0000 0078

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC[3:0]				RANSRC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **LPMDIS**: V_{DDIO1} I/Os Compensation cell low power mode disable.
- 0: IO compensation is disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.
- 1: IO compensation is not disabled by hardware when the system enters Stop, LP-Stop and LPLV-stop modes.

Bit 9 **CS**: V_{DDIO1} I/O code selection

This bit selects the code to be applied for the compensation cell of I/Os supplied by V_{DDIO1}.
- 0: V_{DDIO1} I/O code from the cell (available in SYSCFG_VDDIO1CCSR)
- 1: V_{DDIO1} I/O code from RANSRC[3:0] and RAPSRC[3:0] values

Note: CS = 0 is not taken into account until READY = 1.This means that whenever CS, RANSRC[3:0] and RAPSRC[3:0] are used for I/O compensation when the compensation cell is not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: V_{DDIO1} I/O compensation cell enable

This bit enables the compensation cell of the I/Os supplied by V_{DDIO1}.
- 0: V_{DDIO1} I/O compensation cell disabled
- 1: V_{DDIO1} I/O compensation cell enabled

Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. Similarly, the HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: PMOS compensation code of the I/Os supplied by V_{DDIO1}

These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

Bits 3:0 **RANSRC[3:0]**: NMOS compensation code of the I/Os supplied by V_{DDIO1}

These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit is set.

23.2.51 SYSCFG VDDIO1 compensation cell status register (SYSCFG_VDDIO1CCSR)

Address offset: 0x4024

Reset value: 0x0000 0000

Reset by system reset.

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: V_{DDIO1} I/O compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DDIO1}.

0: V_{DDIO1} I/O compensation cell not ready

1: V_{DDIO1} I/O compensation cell ready

Note: The HSI clock is required for the compensation cell to work properly. The compensation cell READY bit is not set if the HSI clock is not enabled.

Bits 7:4 **APSRC[3:0]**: PMOS compensation value of the I/Os supplied by V_{DDIO1}

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO1CCCR is reset, and the READY bit is set.

Bits 3:0 **ANSRC[3:0]**: NMOS compensation value of the I/Os supplied by V_{DDIO1}

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS in SYSCFG_VDDIO1CCCR is reset, and the READY bit is set.

23.2.52 SYSCFG control timer break register (SYSCFG_CBR)

Address offset: 0x4400

Reset value: 0x0000 0000

Reset by system reset.

Secure privileged write access only. Write only by trusted domain CID. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAML	BKRAML	PVDL	Res.	CLL
											rwo	rwo	rwo		rwo

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAML**: RETRAM double ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the RETRAM double ECC error signal connection to TIM1/8/15/16/17/20 break inputs.

0: RETRAM double ECC error signal disconnected from TIM1/8/15/16/17/20 break inputs

1: RETRAM double ECC error signal connected to TIM1/8/15/16/17/20 break inputs

Bit 3 **BKRAML**: BKPSRAM double ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the BKPSRAM double ECC error signal connection to TIM1/8/15/16/17/20 break inputs.

0: BKPSRAM double ECC error signal disconnected from TIM1/8/15/16/17/20 break inputs

1: BKPSRAM double ECC error signal connected to TIM1/8/15/16/17/20 break inputs

Bit 2 **PVDL**: PVD lock enable

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/8/15/16/17 break input, as well as th PVDEN in PWR_CR3.

0: PVD interrupt disconnected from TIM1/8/15/16/17 break input. PVDEN bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17 break input. PVDEN bits are read only.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CLL**: Cortex-M33 LOCKUP (HardFault) output enable

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of Cortex-M33 LOCKUP (HardFault) output to TIM1/8/15/16/17 break input.

0: Cortex-M33 LOCKUP output disconnected from TIM1/8/15/16/17 break inputs

1: Cortex-M33 LOCKUP output connected to TIM1/8/15/16/17 break inputs

23.2.53 SYSCFG HDP USB debug control register (SYSCFG_HDPUSBDBGCR)

Address offset: 0x5400

Reset value: 0x0000 0000

Reset by system reset.

Write access according to HDP RIFSC configuration. Debug domain can also access if the feature is activated. Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELUSBDBG
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SELUSBDBG**: Selects debug signals to be routed to the Hardware Debug Port

0: debug signals from the USB Host High Speed port are routed to the HDP

1: debug signals from the USB OTG High Speed port are routed to the HDP

23.2.54 SYSCFG device ID register (SYSCFG_IDC)

Address offset: 0x6400

Reset value: 0xXXXX 6503

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Device revision

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x503: STM32MP21xx

23.2.55 SYSCFG version register (SYSCFG_VERR)

Address offset: 0x7FF4

Reset value: 0x0000 0010

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: SYSCFG major revision
The major revision is 1.

Bits 3:0 **MINREV[3:0]**: SYSCFG minor revision
The minor revision is 0.

23.2.56 SYSCFG identification register (SYSCFG_IPIDR)

Address offset: 0x7FF8

Reset value: 0x0003 0001

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SYSCFG identifier

23.2.57 SYSCFG size identification register (SYSCFG_SIDR)

Address offset: 0x7FFC

Reset value: 0xA3C5 DD40

Any read access is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits return the size of the memory region allocated to SYSCFG registers (64-Kbyte decoding space).

23.2.58 SYSCFG register map

Table 172. SYSCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**SYSCFG_BOOTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT3	BOOT2	BOOT1	BOOT0
	Reset value																													X	X	X	X
0x004	**SYSCFG_BOOTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT3_PD	BOOT2_PD	BOOT1_PD	BOOT0_PD
	Reset value																													0	0	0	0
0x008-0x3FC	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x400	**SYSCFG_DLYBSD1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL [5:0]						EN
	Reset value										1	0	0	0	0	0	0										0	0	0	0	0	0	0
0x404	**SYSCFG_DLYBSD1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
	Reset value																															1	0
0x408	**SYSCFG_SDMMC1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1_VSEL
	Reset value																																0
0x40C	**SYSCFG_SDMMC1ARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
	Reset value																																0
0x40C-0x7FC	Reserved	Reserved																															
0x800	**SYSCFG_DLYBSD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL[5:0]						EN
	Reset value										1	0	0	0	0	0	0										0	0	0	0	0	0	0
0x804	**SYSCFG_DLYBSD2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
	Reset value																															1	0
0x808	**SYSCFG_SDMMC2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2_VSEL
	Reset value																																0

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80C	**SYSCFG_SDMMC2ARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
	Reset value																																0
0x80C - 0xBFC	Reserved	Reserved																															
0xC00	**SYSCFG_DLYBSD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTIGLITCH_EN	BYP_CMD[4:0]					BYP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL[5:0]						EN
	Reset value										1	0	0	0	0	0	0										0	0	0	0	0	0	0
0xC04	**SYSCFG_DLYBSD3SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TAP_SEL_ACK	LOCK
	Reset value																															1	0
0xC08	Reserved	Reserved																															
0xC0C	**SYSCFG_SDMMC3ARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
	Reset value																																0
0xC08 - 0xFFC	Reserved	Reserved																															
0x1000	**SYSCFG_DLYBOS1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYP_CMD [4:0]					BYP_EN	Res.	Res.	Res.	TX_TAP_SEL [5:0]						RX_TAP_SEL[5:0]						EN
	Reset value											0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x1004	**SYSCFG_DLYBOS1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TAP_SEL_ACK	RX_TAP_SEL_ACK	LOCK
	Reset value																														1	1	0
0x1008 - 0x1800	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1804	**SYSCFG_ POTTAMPRSTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POTTAMPRESETMASK
	Reset value																																0
0x1808 - 0x1BFC	Reserved	Reserved																															
0x1C00	**SYSCFG_M33SSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKNSMPU	LOCKSMPU	LOCKNSVTOR	LOCKSVTAIRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IT_EN[5:0]					
	Reset value												0	0	0	0	0											0	0	0	0	0	0
0x1C04- 0x1FFC	Reserved	Reserved																															
0x2000	**SYSCFG_ICNQPCR1**	SDMMC3_QOS [3:0]				SDMMC2_QOS [3:0]				SDMMC1_QOS [3:0]				HPDMA3_QOS [3:0]				HPDMA2_QOS [3:0]				HPDMA1_QOS [3:0]				TRACE_QOS [3:0]				A35_QOS [3:0]			
	Reset value	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0	0	1	1	1	1	1	0
0x2004	**SYSCFG_ICNQPCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDC_QOS [3:0]				DCMIPP_QOS [3:0]				Res.	Res.	Res.	Res.	OTG_QOS [3:0]				USBH_QOS [3:0]			
	Reset value													1	0	0	1	1	0	0	0					1	0	1	0	1	0	1	0
0x2008	**SYSCFG_ICNEWRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTG_EARLY_WR_RSP_ENABLE	Res.	SDMMC3_EARLY_WR_RSP_ENABLE	SDMMC2_EARLY_WR_RSP_ENABLE	SDMMC1_EARLY_WR_RSP_ENABLE	USBH_EARLY_WR_RSP_ENABLE	
	Reset value																										0		0	0	0	0	
0x200C - 0x2010	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2014	**SYSCFG_ ICNE2EBWRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG_HPDMA_BW	CFG_CPU1_BW	RESERVED1	CFG_ALL_TGT [2:0]			Res.	CFG_MODEL[1:0]		CFG_ENABLE
	Reset value																							0	0	0	1	0	1		0	1	0
0x2018	**SYSCFG_SAFERSTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG_ENABLE
	Reset value																																0
0x201C	Reserved	Reserved																															
0x2020	**SYSCFG_ ICNETHBWLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
	Reset value																	0	0	0	1	0	0	0	0							1	
0x2024	Reserved	Reserved																															
0x2028	**SYSCFG_ ICNCPU1BWLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
	Reset value																	0	0	0	1	1	0	0	0							1	
0x202C	**SYSCFG_ ICNLTDCBWLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
	Reset value																	0	0	0	1	1	1	1	0							1	
0x2030	**SYSCFG_ ICNDCMIPPBWLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
	Reset value																	0	0	1	0	0	1	0	0							1	
0x2034	Reserved	Reserved																															
0x2038	**SYSCFG_ ICNHPDMABWLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXABW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	BWLEN	Res.
	Reset value																	0	1	0	0	1	0	0	0							1	
0x203C-0x234C	Reserved	Reserved																															
0x2050	**SYSCFG_ HPDMAARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA3AREN	HPDMA2AREN	HPDMA1AREN
	Reset value																														0	0	0
0x2054-0x23FC	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2400	**SYSCFG_ USB2PHY1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1SEL[2:0]			Res.	USB2PHY1CMN	AUTORSMENB1	RETENABLEN1
	Reset value																										0	0	1		1	0	1
0x2404	**SYSCFG_ USB2PHY1BCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H_FORCEBCMODE	H_VDMSRCEN	H_CDPDETEN	H_CDPEN
	Reset value																													0	0	0	0
0x2408	**SYSCFG_ USB2PHY1BCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSVMINUS	FSVPLUS	CHGDET
	Reset value																														0	0	0
0x240C	**SYSCFG_ USB2PHY1TRIM1CR**	Res.	USB2PHY1TXRESTUNE [1:0]		USB2PHY1TXRISETUNE [1:0]		USB2PHY1TXVREFTUNE [3:0]				USB2PHY1TXFSLSTUNE [3:0]				USB2PHY1TXHSXVTUNE [1:0]		USB2PHY1OTGTUNE [2:0]			USB2PHY1VDATREFTUNE [1:0]		USB2PHY1SQRXTUNE [2:0]			USB2PHY1COMPDISTUNE [2:0]			USB2PHY1PLLPTUNE [3:0]				USB2PHY1PLLITUNE [1:0]		
	Reset value		0	1	0	1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0
0x2410	**SYSCFG_ USB2PHY1TRIM2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY1TXPREEMPPULSETUNE	USB2PHY1TXPREEMPAMPTUNE [1:0]	
	Reset value																														0	0	0
0x2414 - 0x241C	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2420	**SYSCFG_USBHCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBH_VBUSEN_POLARITY	USBH_OVRCUR_POLARITY
	Reset value																															0	0
0x2424	**SYSCFG_USBHARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
	Reset value																																0
0x2428 - 0x27FC	Reserved	Reserved																															
0x2800	**SYSCFG_ USB2PHY2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2SEL [2:0]			Res.	USB2PHY2CMN	AUTORSMENB2	RETENABLEN2
	Reset value																										0	0	1		1	0	1
0x2804	Reserved	Reserved																															
0x2808	**SYSCFG_ USB2PHY2TRIM1CR**	Res.	USB2PHY2TXRESTUNE [1:0]		USB2PHY2TXRISETUNE [1:0]		USB2PHY2TXVREFTUNE [3:0]				USB2PHY2TXFSLSTUNE [3:0]				USB2PHY2TXHSXVTUNE [1:0]		USB2PHY2OTGTUNE [2:0]			USB2PHY2VDATREFTUNE [1:0]		USB2PHY2SQRXTUNE [2:0]			USB2PHY2COMPDISTUNE [2:0]			USB2PHY2PLLPTUNE [3:0]				USB2PHY2PLLITUNE [1:0]	
	Reset value		0	1	0	1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0
0x280C	**SYSCFG_ USB2PHY2TRIM2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2PHY2TXPREEMPPULSETUNE	USB2PHY2TXPREEMPAMPTUNE [1:0]	
	Reset value																														0	0	0
0x2810 - 0x2820	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2824	**SYSCFG_ OTGARCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AREN
	Reset value																																0
0x2828 - 0x2FFC	Reserved	Reserved																															
0x3000	**SYSCFG_ETH1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1_SEL[2:0]			Res.	Res.	ETH1_CLK_SEL	ETH1_REF_CLK_SEL
	Reset value																										0	0	0			0	0
0x3004 - 0x300C	Reserved	Reserved																															
0x3010	**SYSCFG_ETH1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1_PWR_DOWN_ACK	ETH1_TX_CLK_GATING_CTRL
	Reset value																															0	0
0x3014 - 0x33FC	Reserved	Reserved																															
0x3400	**SYSCFG_ETH2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH2_SEL[2:0]			Res.	Res.	ETH2_CLK_SEL	ETH2_REF_CLK_SEL
	Reset value																										0	0	0			0	0
0x3404 - 0x340C	Reserved	Reserved																															

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3410	**SYSCFG_ETH2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH2_PWR_DOWN_ACK	ETH2_TX_CLK_GATING_CTRL
	Reset value																															0	0
0x3414 - 0x3FFC	Reserved	Reserved																															
0x4000	**SYSCFG_ VDDIO3CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																						0	0	0	0	1	1	1	1	0	0	0
0x4004	**SYSCFG_ VDDIO3CCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x4008 - 0x400C	Reserved	Reserved																															
0x4010	**SYSCFG_VDDCCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																						0	0	0	0	1	1	1	1	0	0	0
0x4014	**SYSCFG_VDDCCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x4018	**SYSCFG_ VDDIO2CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																						0	0	0	0	1	1	1	1	0	0	0
0x401C	**SYSCFG_ VDDIO2CCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x4020	**SYSCFG_ VDDIO1CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMDIS	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																						0	0	0	0	1	1	1	1	0	0	0
0x4024	**SYSCFG_ VDDIO1CCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0

Table 172. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4028 - 0x43FC	Reserved	Reserved																															
0x4400	**SYSCFG_CBR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAM	BKRAML	PVDL	Res.	CLL
	Reset value																												0	0	0		0
0x4404 - 0x63FC	Reserved	Reserved																															
0x5400	**SYSCFG_ HDPUSBDBGCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELUSBDBG
	Reset value																																0
0x6400	**SYSCFG_IDC**	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					0	1	0	1	0	0	0	0	0	0	1	1
0x6404 - 0x7FF0	Reserved	Reserved																															
0x7FF4	**SYSCFG_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x7FF8	**SYSCFG_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x7FFC	**SYSCFG_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

24 Peripheral interconnects

24.1 Introduction

Several peripherals have direct interconnections between them. This allows autonomous communication and synchronization between peripherals, saving processor resources and reducing power supply consumption.

In addition, these hardware connections remove software latency and allow the design of a predictable system.

This also saves pins and GPIOs.

Depending on the peripheral, these interconnections can operate in Run, CSleep or Stop modes.

24.2 Connection overview

There are several types of connections, as listed below.

24.2.1 Asynchronous connections (A)

The source output signal is sampled by the destination clock, leading to the possible introduction of jitter in the latency between the source output event and the destination event detection.

24.2.2 Synchronous connections (S)

Both source and destination are synchronous (they run on the same clock), and the latency from the source to the destination is deterministic. No jitter is introduced.

24.2.3 Break/fault connection for TIM outputs (B)

The source output signal disables the timer outputs through a pure combinatorial logic path, without any latency.

24.2.4 Peripherals interconnect matrix

Table 173. Peripherals interconnect matrix

| - | Destination | | |

Table 173. Peripherals interconnect matrix (continued)

| - | Source | Destination | | |

Table 173. Peripherals interconnect matrix (continued)

| - | Destination | | |

Table 174. Peripherals interconnect matrix details

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB2	TIM1	TRGO	EXT0	ADC1	AHB2
APB2	TIM1	TRGO2	EXT1		
APB2	TIM8	TRGO	EXT2		
APB2	TIM8	TRGO2	EXT3		
AHB4	EXTI1	EXTI12	EXT4		
APB1	TIM2	TRGO	EXT6		
APB1	TIM3	TRGO	EXT7		
APB1	TIM4	TRGO	EXT8		
APB1	TIM5	TRGO	EXT9		
APB1	TIM6	TRGO	EXT10		
APB2	TIM15	TRGO	EXT11		
APB2	TIM1	OC1	EXT12		
APB2	TIM1	OC2	EXT13		
APB2	TIM1	OC3	EXT14		
APB1	TIM2	OC2	EXT18		
APB1	TIM3	OC4	EXT19		
APB1	TIM4	OC4	EXT20		
APB1	TIM5	OC1	EXT21		
APB1	TIM12	OC1	EXT22		
AHB4	EXTI2	EXTI11	EXT23		
APB1	LPTIM1	CH1	EXT24		
APB1	LPTIM2	CH1	EXT25		
APB5	LPTIM3	CH1	EXT26		
APB5	LPTIM4	CH1	EXT27		
APB5	LPTIM5	OUT	EXT28		
APB2	TIM1	TRGO	JEXT0		
APB2	TIM1	TRGO2	JEXT1		
APB2	TIM8	TRGO	JEXT2		
APB2	TIM8	TRGO2	JEXT3		
AHB4	EXTI1	EXTI14	JEXT4		
APB1	TIM2	TRGO	JEXT6		
APB1	TIM3	TRGO	JEXT7		
APB1	TIM4	TRGO	JEXT8		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB1	TIM5	TRGO	JEXT9	ADC1 (cont'd)	AHB2 (cont'd)
APB1	TIM6	TRGO	JEXT10		
APB1	TIM7	TRGO	JEXT11		
APB2	TIM15	TRGO	JEXT12		
APB2	TIM16	OC1	JEXT13		
APB2	TIM1	OC3	JEXT14		
APB2	TIM8	OC1	JEXT15		
APB1	TIM2	OC1	JEXT17		
APB1	TIM2	OC3	JEXT18		
APB1	TIM3	OC1	JEXT19		
APB1	TIM4	OC1	JEXT20		
APB1	TIM5	OC3	JEXT21		
APB1	TIM12	OC1	JEXT22		
AHB4	EXTI2	EXTI15	JEXT23		
APB1	LPTIM1	CH2	JEXT24		
APB1	LPTIM2	CH2	JEXT25		
APB5	LPTIM3	CH1	JEXT26		
APB5	LPTIM4	CH1	JEXT27		
APB5	LPTIM5	OUT	JEXT28		
APB2	TIM1	TRGO	EXT0	ADC2	AHB2
APB2	TIM1	TRGO2	EXT1		
APB2	TIM8	TRGO	EXT2		
APB2	TIM8	TRGO2	EXT3		
AHB4	EXTI1	EXTI6	EXT4		
APB1	TIM2	TRGO	EXT6		
APB1	TIM3	TRGO	EXT7		
APB1	TIM4	TRGO	EXT8		
APB1	TIM5	TRGO	EXT9		
APB1	TIM6	TRGO	EXT10		
APB1	TIM7	TRGO	EXT11		
APB2	TIM15	TRGO	EXT12		
APB2	TIM17	OC1	EXT13		
APB2	TIM1	OC3	EXT14		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB2	TIM8	OC1	EXT15	ADC2 (cont'd)	AHB2 (cont'd)
APB1	TIM2	OC1	EXT17		
APB1	TIM2	OC3	EXT18		
APB1	TIM3	OC1	EXT19		
APB1	TIM4	OC1	EXT20		
APB1	TIM5	OC3	EXT21		
APB1	TIM12	OC1	EXT22		
AHB4	EXTI2	EXTI2	EXT23		
APB1	LPTIM1	CH1	EXT24		
APB1	LPTIM2	CH1	EXT25		
APB5	LPTIM3	CH1	EXT26		
APB5	LPTIM4	CH1	EXT27		
APB5	LPTIM5	OUT	EXT28		
APB2	TIM1	TRGO	JEXT0		
APB2	TIM1	TRGO2	JEXT1		
APB2	TIM8	TRGO	JEXT2		
APB2	TIM8	TRGO2	JEXT3		
AHB4	EXTI1	EXTI7	JEXT4		
APB1	TIM2	TRGO	JEXT6		
APB1	TIM3	TRGO	JEXT7		
APB1	TIM4	TRGO	JEXT8		
APB1	TIM5	TRGO	JEXT9		
APB1	TIM6	TRGO	JEXT10		
APB1	TIM7	TRGO	JEXT11		
APB2	TIM15	TRGO	JEXT12		
APB2	TIM1	OC3	JEXT13		
APB2	TIM1	OC4	JEXT14		
APB2	TIM8	OC2	JEXT15		
APB2	TIM8	OC4	JEXT16		
APB1	TIM4	OC3	JEXT18		
APB1	TIM4	OC4	JEXT19		
APB1	TIM5	OC1	JEXT20		
APB1	TIM12	OC2	JEXT21		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB2	TIM15	OC1	JEXT22	ADC2 (cont'd)	AHB2 (cont'd)
AHB4	EXTI2	EXTI3	JEXT23		
APB1	LPTIM1	CH2	JEXT24		
APB1	LPTIM2	CH2	JEXT25		
APB5	LPTIM3	CH1	JEXT26		
APB5	LPTIM4	CH1	JEXT27		
APB5	LPTIM5	OUT	JEXT28		
APB1	LPTIM1	CH2	TRG1	DTS	APB3
APB1	LPTIM2	CH2	TRG2		
APB5	LPTIM3	CH1	TRG3		
AHB4	EXTI2	EXTI13	TRG4		
APB1	TIM2	TRGO	PTP0	ETH1	AHB5
APB1	TIM3	TRGO	PTP1		
-	AFMUX	ETH1_PTP_AUX_TS	PTP2		
APB1	TIM2	TRGO	PTP0	ETH2	AHB5
APB1	TIM3	TRGO	PTP1		
-	AFMUX	ETH2_PTP_AUX_TS	PTP2		
APB1	TIM2	TRGO	SWT0	FDCAN1	APB2
APB1	TIM3	TRGO	SWT1		
AHB5	ETH1	PPS_OUT1	SWT2		
APB1	TIM2	TRGO	EVT0		
APB1	TIM3	TRGO	EVT1		
AHB5	ETH1	PPS_OUT1	EVT2		
APB1	TIM3	CNT[15:0]	EXT_TS[15:0]		
AHB5	ETH2	PPS_OUT1	SWT3		
AHB5	ETH2	PPS_OUT1	EVT3		
AHB4	EXTI2	EXTI4	TRIG4	I2C1	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10	I2C1	APB1
APB5	RTC	WUT	TRIG11		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB4	EXTI1	EXTI6	TRIG12	I2C1 (cont'd)	APB1 (cont'd)
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	I2C2	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	I2C3	APB5
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	I3C1 (cont'd)	APB1 (cont'd)
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB4	EXTI1	EXTI6	TRIG12	I3C1	APB1
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	I3C2	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	I3C3	APB5
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
-	AFMUX	LPTIM1_ETR	ETR0	LPTIM1	APB1
APB5	RTC	ALRA	ETR1		
APB5	RTC	ALRB	ETR2		
APB5	TAMP	TAMP1	ETR3		
APB5	TAMP	TAMP2	ETR4		
APB2	SAI2	FS_A	ETR6		
APB2	SAI2	FS_B	ETR7		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
-	AFMUX	LPTIM1_IN1	IN1_0	LPTIM1 (cont'd)	APB1 (cont'd)
-	AFMUX	LPTIM1_CH1	IC1_0		
-	AFMUX	LPTIM1_IN2	IN2_0		
-	AFMUX	LPTIM1_CH2	IC2_0		
AHB4	RCC	HSI/1024	IC2_1		
AHB4	RCC	HSI/1024	ETR5		
AHB4	RCC	MSI	IC2_2		
AHB4	RCC	HSE	IN1_1		
APB1	I3C1	IBIACK	IC1_1		
APB1	I3C2	IBIACK	IC1_2		
APB5	I3C3	IBIACK	IC1_3		
-	AFMUX	LPTIM2_ETR	ETR0	LPTIM2	APB1
APB5	RTC	ALRA	ETR1		
APB5	RTC	ALRB	ETR2		
APB5	TAMP	TAMP1	ETR3		
APB5	TAMP	TAMP2	ETR4		
APB5	TAMP	TAMP3	ETR5		
APB2	SAI4	FS_A	ETR6		
APB2	SAI4	FS_B	ETR7		
-	AFMUX	LPTIM2_IN1	IN1_0		
-	AFMUX	LPTIM2_CH1	IC1_0		
-	AFMUX	LPTIM2_IN2	IN2_0		
-	AFMUX	LPTIM2_CH2	IC2_0		
APB1	I3C1	IBIACK	IC1_1		
APB1	I3C2	IBIACK	IC1_2		
APB5	I3C3	IBIACK	IC1_3		
-	AFMUX	LPTIM3_ETR	ETR0	LPTIM3	APB5
APB5	RTC	ALRA	ETR1		
APB5	RTC	ALRB	ETR2		
APB5	TAMP	TAMP1	ETR3		
APB5	TAMP	TAMP2	ETR4		
APB5	TAMP	TAMP3	ETR5		
APB5	LPTIM4	CH1	ETR6		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB5	LPTIM5	OUT	ETR7	LPTIM3 (cont'd)	APB5 (cont'd)
-	AFMUX	LPTIM3_IN1	IN1_0		
-	AFMUX	LPTIM3_CH1	IC1_0		
-	AFMUX	LPTIM3_CH2	IC2_0		
AHB4	RCC	MSI	IC1_1		
APB1	I3C2	IBIACK	IC1_2		
APB5	I3C3	IBIACK	IC1_3		
-	AFMUX	LPTIM4_ETR	ETR0	LPTIM4	APB5
APB5	RTC	ALRA	ETR1		
APB5	RTC	ALRB	ETR2		
APB5	TAMP	TAMP1	ETR3		
APB5	TAMP	TAMP2	ETR4		
APB5	TAMP	TAMP3	ETR5		
APB5	LPTIM3	CH1	ETR6		
APB5	LPTIM5	OUT	ETR7		
-	AFMUX	LPTIM4_IN1	IN1_0		
-	AFMUX	LPTIM4_CH1	IC1_0		
-	AFMUX	LPTIM4_CH2	IC2_0		
AHB4	RCC	LSI	IC1_1		
AHB4	RCC	LSE	IC1_2		
-	AFMUX	LPTIM5_ETR	ETR0	LPTIM5	APB5
APB5	RTC	ALRA	ETR1		
APB5	RTC	ALRB	ETR2		
APB5	TAMP	TAMP1	ETR3		
APB5	TAMP	TAMP2	ETR4		
APB5	TAMP	TAMP3	ETR5		
APB5	LPTIM3	CH1	ETR6		
APB5	LPTIM4	CH1	ETR7		
-	AFMUX	LPTIM5_IN1	IN1_0		
AHB4	EXTI2	EXTI4	TRIG4	LPUART1	APB5
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB5	LPTIM5	OUT	TRIG9	LPUART1 (cont'd)	APB5 (cont'd)
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
APB2	TIM1	TRGO2	TRG0	MDF1	AHB2
APB2	TIM8	TRGO2	TRG1		
APB2	TIM16	OC	TRG3		
APB1	TIM6	TRGO	TRG4		
APB1	TIM7	TRGO	TRG5		
AHB4	EXTI2	EXTI11	TRG6		
AHB4	EXTI2	EXTI15	TRG7		
APB1	LPTIM1	CH1	TRG8		
APB1	LPTIM2	CH1	TRG9		
APB5	LPTIM3	CH1	TRG10		
AHB2	ADC1	DATA	ADC1DAT		
AHB2	ADC2	DATA	ADC2DAT		
APB2	SAI2	SYNCOUT_FS	SYNCIN1_FS	SAI1	APB2
APB2	SAI2	SYNCOUT_SCK	SYNCIN1_SCK		
APB2	SAI3	SYNCOUT_FS	SYNCIN2_FS		
APB2	SAI3	SYNCOUT_SCK	SYNCIN2_SCK		
APB2	SAI4	SYNCOUT_FS	SYNCIN3_FS		
APB2	SAI4	SYNCOUT_SCK	SYNCIN3_SCK		
APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI2	APB2
APB2	SAI1	SYNCOUT_SCK	SYNCIN0_SCK		
APB2	SAI3	SYNCOUT_FS	SYNCIN2_FS		
APB2	SAI3	SYNCOUT_SCK	SYNCIN2_SCK		
APB2	SAI4	SYNCOUT_FS	SYNCIN3_FS	SAI2	APB2
APB2	SAI4	SYNCOUT_SCK	SYNCIN3_SCK		
APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI3	APB2
APB2	SAI1	SYNCOUT_SCK	SYNCIN0_SCK		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB2	SAI2	SYNCOUT_FS	SYNCIN1_FS	SAI3 (cont'd)	APB2 (cont'd)
APB2	SAI2	SYNCOUT_SCK	SYNCIN1_SCK		
APB2	SAI4	SYNCOUT_FS	SYNCIN3_FS		
APB2	SAI4	SYNCOUT_SCK	SYNCIN3_SCK		
APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI4	APB2
APB2	SAI1	SYNCOUT_SCK	SYNCIN0_SCK		
APB2	SAI2	SYNCOUT_FS	SYNCIN1_FS		
APB2	SAI2	SYNCOUT_SCK	SYNCIN1_SCK		
APB2	SAI3	SYNCOUT_FS	SYNCIN2_FS		
APB2	SAI3	SYNCOUT_SCK	SYNCIN2_SCK		
AHB4	EXTI2	EXTI4	TRIG4	SPI1	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	SPI2	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13	SPI2	APB1
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB4	EXTI2	EXTI4	TRIG4	SPI3	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	SPI4	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	SPI5	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13	SPI5	APB2
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB4	EXTI2	EXTI4	TRIG4	SPI6	APB5
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
APB1	LPTIM1	CH1	ITAMP4	TAMP	APB5
AHB2	ADC2	AWD1	ITAMP7		
AHB2	ADC2	AWD2	ITAMP12		
AHB2	ADC2	AWD3	ITAMP13		
APB1	TIM2	TRGO	ITR1	TIM1	APB2
APB1	TIM3	TRGO	ITR2		
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
-	AFMUX	TIM1_ETR	ETR0		
AHB2	ADC1	AWD1	ETR5		
AHB2	ADC1	AWD2	ETR6		
AHB2	ADC1	AWD3	ETR7		
AHB2	ADC2	AWD1	ETR8		
AHB2	ADC2	AWD2	ETR9		
AHB2	ADC2	AWD3	ETR10		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
-	AFMUX	TIM1_CH1	TI1_0	TIM1 (cont'd)	APB2 (cont'd)
-	AFMUX	TIM1_CH2	TI2_0		
-	AFMUX	TIM1_CH3	TI3_0		
-	AFMUX	TIM1_CH4	TI4_0		
-	AFMUX	TIM1_BKIN	BRK_0		
AHB2	MDF1	BRK0	BRK_8		
-	AFMUX	TIM1_BKIN2	BRK2_0		
AHB2	MDF1	BRK1	BRK2_8		
-	SOC	SYSBRK	BRK_ACTH		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
-	AFMUX	TIM10_CH1	TI1_0	TIM10	APB1
-	AFMUX	TIM11_CH1	TI1_0	TIM11	APB1
APB2	TIM1	TRGO	ITR0	TIM12	APB1
APB1	TIM2	TRGO	ITR1		
APB1	TIM3	TRGO	ITR2		
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
-	AFMUX	TIM12_CH1	TI1_0		
-	AFMUX	TIM12_CH2	TI2_0		
AHB4	RCC	MSI/256	TI1_1		
AHB4	RCC	HSI/1024	TI1_2		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
-	AFMUX	TIM13_CH1	TI1_0	TIM13	APB1
-	AFMUX	TIM14_CH1	TI1_0	TIM14	APB1
-	AFMUX	TIM15_CH1	TI1_0	TIM15	APB2

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
-	AFMUX	TIM2_CH1	TI1_1	TIM15 (cont'd)	APB2 (cont'd)
-	AFMUX	TIM3_CH1	TI1_2		
-	AFMUX	TIM4_CH1	TI1_3		
AHB4	RCC	LSE	TI1_4		
AHB4	RCC	MSI	TI1_5		
AHB4	RCC	MCO2	TI1_6		
-	AFMUX	TIM15_CH2	TI2_0		
-	AFMUX	TIM2_CH2	TI2_1		
-	AFMUX	TIM3_CH2	TI2_2		
-	AFMUX	TIM4_CH2	TI2_3		
-	AFMUX	TIM15_BKIN	BRK_0		
AHB2	MDF1	BRK0	BRK_8		
-	SOC	SYSBRK	BRK_ACTH		
AHB4	RCC	MSI/256	TI1_7		
AHB4	RCC	HSI/1024	TI1_8		
APB2	TIM1	TRGO	ITR0		
APB1	TIM2	TRGO	ITR1		
APB1	TIM3	TRGO	ITR2		
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
APB5	I3C3	IBIACK	TI1_9		
-	AFMUX	TIM16_CH1	TI1_0	TIM16	APB2
AHB4	RCC	LSI	TI1_1		
AHB4	RCC	LSE	TI1_2		
APB5	RTC	WUT	TI1_3		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
-	AFMUX	TIM16_BKIN	BRK_0	TIM16 (cont'd)	APB2 (cont'd)
AHB2	MDF1	BRK1	BRK_8		
-	SOC	SYSBRK	BRK_ACTH		
APB1	I3C1	IBIACK	TI1_7		
APB1	I3C2	IBIACK	TI1_8		
APB5	I3C3	IBIACK	TI1_9		
-	AFMUX	TIM17_CH1	TI1_0	TIM17	APB2
APB1	SPDIFRX	FS	TI1_1		
AHB4	RCC	HSE	TI1_2		
AHB4	RCC	MCO1	TI1_3		
-	AFMUX	TIM17_BKIN	BRK_0		
AHB2	MDF1	BRK2	BRK_8		
-	SOC	SYSBRK	BRK_ACTH		
APB1	I3C1	IBIACK	TI1_7		
APB1	I3C2	IBIACK	TI1_8		
APB5	I3C3	IBIACK	TI1_9		
-	AFMUX	TIM2_ETR	ETR0	TIM2	APB1
-	AFMUX	TIM3_ETR	ETR2		
-	AFMUX	TIM4_ETR	ETR3		
-	AFMUX	TIM5_ETR	ETR4		
AHB4	RCC	LSE	ETR5		
APB2	SAI1	FS_A	ETR6		
APB2	SAI1	FS_B	ETR7		
AHB5	ETH1	PPS_OUT1	ETR8		
AHB5	ETH2	PPS_OUT1	ETR9		
-	AFMUX	TIM2_CH1	TI1_0		
-	AFMUX	TIM2_CH2	TI2_0		
-	AFMUX	TIM2_CH3	TI3_0		
-	AFMUX	TIM2_CH4	TI4_0		
AHB5	ETH1	PPS_OUT1	TI1_1		
AHB5	ETH2	PPS_OUT1	TI2_1		
APB1	TIM12	TRGO	TI4_2		
APB2	TIM1	TRGO	ITR0		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB1	TIM3	TRGO	ITR2	TIM2 (cont'd)	APB1 (cont'd)
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
APB2	TIM1	TRGO	ITR0	TIM3	APB1
APB1	TIM2	TRGO	ITR1		
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
-	AFMUX	TIM3_ETR	ETR0		
-	AFMUX	TIM2_ETR	ETR1		
-	AFMUX	TIM4_ETR	ETR3		
-	AFMUX	TIM5_ETR	ETR4		
AHB5	ETH1	PPS_OUT1	ETR6		
AHB5	ETH2	PPS_OUT1	ETR7		
-	AFMUX	TIM3_CH1	TI1_0		
-	AFMUX	TIM3_CH2	TI2_0		
-	AFMUX	TIM3_CH3	TI3_0		
-	AFMUX	TIM3_CH4	TI4_0		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB5	ETH1	PPS_OUT1	TI1_1	TIM3 (cont'd)	APB1 (cont'd)
AHB5	ETH2	PPS_OUT1	TI2_1		
APB1	TIM12	TRGO	TI4_2		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
-	AFMUX	TIM4_ETR	ETR0	TIM4	APB1
-	AFMUX	TIM2_ETR	ETR1		
-	AFMUX	TIM3_ETR	ETR2		
-	AFMUX	TIM5_ETR	ETR4		
APB2	TIM1	TRGO	ITR0		
APB1	TIM2	TRGO	ITR1		
APB1	TIM3	TRGO	ITR2		
APB1	TIM5	TRGO	ITR4		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
-	AFMUX	TIM4_CH1	TI1_0		
-	AFMUX	TIM4_CH2	TI2_0		
-	AFMUX	TIM4_CH3	TI2_0		
-	AFMUX	TIM4_CH4	TI2_0		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
APB2	TIM1	TRGO	ITR0	TIM5	APB1
APB1	TIM2	TRGO	ITR1		
APB1	TIM3	TRGO	ITR2		
APB1	TIM4	TRGO	ITR3		
APB2	TIM8	TRGO	ITR5		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB1	TIM14	OC	ITR8	TIM5 (cont'd)	APB1 (cont'd)
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
APB2	FDCAN1	SOC	ITR15		
-	AFMUX	TIM5_ETR	ETR0		
-	AFMUX	TIM2_ETR	ETR1		
-	AFMUX	TIM3_ETR	ETR2		
-	AFMUX	TIM4_ETR	ETR3		
APB2	SAI2	FS_A	ETR6		
APB2	SAI2	FS_B	ETR7		
AHB5	USBH	SOF	ETR8		
AHB5	OTG	SOF	ETR9		
APB2	FDCAN1	TMP	TI1_1		
APB2	FDCAN1	RTP	TI1_2		
-	AFMUX	TIM5_CH1	TI1_0		
-	AFMUX	TIM5_CH2	TI2_0		
-	AFMUX	TIM5_CH3	TI3_0		
-	AFMUX	TIM5_CH4	TI4_0		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
APB2	TIM1	TRGO	ITR0	TIM8	APB2
APB1	TIM2	TRGO	ITR1		
APB1	TIM3	TRGO	ITR2		
APB1	TIM4	TRGO	ITR3		
APB1	TIM5	TRGO	ITR4		
APB1	TIM12	TRGO	ITR6		
APB1	TIM13	OC	ITR7		
APB1	TIM14	OC	ITR8		
APB2	TIM15	TRGO	ITR9		
APB2	TIM16	OC	ITR10		
APB2	TIM17	OC	ITR11		
-	AFMUX	TIM8_ETR	ETR0		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
AHB2	ADC1	AWD1	ETR5	TIM8 (cont'd)	APB2 (cont'd)
AHB2	ADC1	AWD2	ETR6		
AHB2	ADC1	AWD3	ETR7		
AHB2	ADC2	AWD1	ETR8		
AHB2	ADC2	AWD2	ETR9		
AHB2	ADC2	AWD3	ETR10		
-	AFMUX	TIM8_CH1	TI1_0		
-	AFMUX	TIM8_CH2	TI2_0		
-	AFMUX	TIM8_CH3	TI3_0		
-	AFMUX	TIM8_CH4	TI4_0		
-	AFMUX	TIM8_BKIN	BRK_0		
AHB2	MDF1	BRK2	BRK_8		
-	AFMUX	TIM8_BKIN2	BRK2_0		
AHB2	MDF1	BRK3	BRK2_8		
-	SOC	SYSBRK	BRK_ACTH		
APB1	TIM10	OC	ITR13		
APB1	TIM11	OC	ITR14		
AHB4	EXTI2	EXTI4	TRIG4	UART4	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	UART5	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB5	RTC	ALRA	TRIG10	UART5 (cont'd)	APB1 (cont'd)
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	UART7	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	USART1	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	USART2	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		

Table 174. Peripherals interconnect matrix details (continued)

Source			Destination		
Bus	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**
APB5	RTC	ALRA	TRIG10	USART2 (cont'd)	APB1 (cont'd)
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	USART3	APB1
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		
AHB4	EXTI2	EXTI4	TRIG4	USART6	APB2
AHB4	EXTI2	EXTI5	TRIG5		
APB5	LPTIM3	CH1	TRIG7		
APB5	LPTIM4	CH1	TRIG8		
APB5	LPTIM5	OUT	TRIG9		
APB5	RTC	ALRA	TRIG10		
APB5	RTC	WUT	TRIG11		
AHB4	EXTI1	EXTI6	TRIG12		
AHB4	EXTI1	EXTI7	TRIG13		
APB1	LPTIM1	CH1	TRIG14		
APB1	LPTIM2	CH1	TRIG15		

25 High-performance direct memory access controller (HPDMA)

25.1 HPDMA introduction

The high-performance direct memory access (HPDMA) controller is a bus master and system peripheral.

The HPDMA is used to perform programmable data transfers between memory-mapped peripherals, and/or memories via linked-lists, upon the control of an off-loaded CPU.

25.2 HPDMA main features

- Single bidirectional AXI master and single bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent HPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 25.3.1*)
  - Intra-channel HPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel HPDMA transfers chaining via programmable HPDMA input triggers connection to HPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level and byte-based padding
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at block level, between successive burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 25.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Selection of programmable HPDMA request and trigger
  - Generation of programmable HPDMA half-transfer and transfer-complete event
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the HPDMA linked-list control registers
  - Channel abort and restart
  - Channel terminate with FIFO draining
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure HPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any HPDMA secure resource (register, bitfield) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged HPDMA transfers, independently at channel level
  - Privileged-aware AHB slave port
- Channel isolation support:
  - Support for compartmented DMA transfers, independently at channel level, via compartment IDs (named CIDs)
  - CID-aware interrupts reporting
  - CID-aware AHB slave port, with integrated semaphores for a concurrent control, from any of the CPUs
- Identification registers, with associated peripheral revision and hardware configuration

25.3 HPDMA implementation

25.3.1 HPDMA channels

A given HPDMA channel x is implemented with the following features and intended use.

To make the best use of the HPDMA performance, the following table lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested HPDMA transfer.

Table 175. Implementation of HPDMA1/2/3 channels

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 7	4	0	Channel x (x = 0 to 7) is implemented with: a FIFO of 32 bytes, 8 words, 4 double-words fixed/contiguously incremented addressing These channels are typically allocated for HPDMA transfers between an APB or AHB peripheral, and SRAM.
x = 8 to 11	5	0	Channel x (x = 8 to 11) is implemented with: a FIFO of 64 bytes, 16 words, 8 double-words fixed/contiguously incremented addressing These channels can be used for HPDMA transfers between a demanding AHB peripheral and SRAM, or with external memories transfers.
x = 12 to 14	6	2	Channel x (x = 12 to 14) is implemented with: a FIFO of 128 bytes, 16 double-words 2D addressing with extended offsets These channels can also be used for data demanding memory-to-memory HPDMA transfers including DDR-SDRAM.
x = 15	7	2	Channel x (x = 15) is implemented with: a FIFO of 256 bytes, 32 double-words 2D addressing with extended offsets These channels can also be used for data demanding memory-to-memory HPDMA transfers including DDR-SDRAM.

25.3.2 HPDMA allowed AXI maximum burst length

For a data transfer, the user has to program:

– the source (and destination) burst length that a given channel x ideally performs, via SBL_1[5:0] (and respectively DBL_1[5:0]) field in HPDMA_CxTR1.

– the allocated master port, AXI or AHB, on which this source (or destination) data transfer must be performed, via SAP (or respectively DAP) field in HPDMA_CxTR1.

Caution: The maximum allowed AXI burst length is restricted to 16:

1. if SAP = 0 (AXI allocated port): the maximum allowed value of SBL_1[5:0] is 15.
2. if DAP = 0 (AXI allocated port): the maximum allowed value of DBL_1[5:0] is 15.

25.3.3 HPDMA in low-power modes

The HPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 176. HPDMA1/2/3 wake-up in low-power modes

Feature	Low-power modes
Wake-up	HPDMA1/2/3 in Sleep and Stop 1 modes

25.3.4 HPDMA requests

An HPDMA request from a peripheral can be assigned to a HPDMA channel x, via REQSEL[7:0] in HPDMA_CxTR2, provided that SWREQ = 0.

The HPDMA requests mapping is specified in the table below.

Table 177. Programmed HPDMA1/2/3 request

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
0	fmc_nand_txrx_dma
1	fmc_nand_bch_dma
2	octospi1_dma
3	Reserved
4	cryp1_in_dma
5	cryp1_out_dma
6	hash1_dma
7	saes_out_dma
8	saes_in_dma
9	usart1_rx_dma
10	usart1_tx_dma
11	usart2_rx_dma
12	usart2_tx_dma
13	usart3_rx_dma
14	usart3_tx_dma
15	uart4_rx_dma
16	uart4_tx_dma
17	uart5_rx_dma
18	uart5_tx_dma
19	usart6_rx_dma
20	usart6_tx_dma
21	uart7_rx_dma
22	uart7_tx_dma
23	i2c1_rx_dma
24	i2c1_tx_dma
25	i2c1_evc_dma
26	i2c2_rx_dma
27	i2c2_tx_dma
28	i2c2_evc_dma
29	i2c3_rx_dma

Table 177. Programmed HPDMA1/2/3 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
30	i2c3_tx_dma
31	i2c3_evc_dma
32	spi1_rx_dma
33	spi1_tx_dma
34	spi2_rx_dma
35	spi2_tx_dma
36	spi3_rx_dma
37	spi3_tx_dma
38	spi4_rx_dma
39	spi4_tx_dma
40	spi5_rx_dma
41	spi5_tx_dma
42	spi6_rx_dma
43	spi6_tx_dma
44	mdf1_flt0_dma
45	mdf1_flt1_dma
46	mdf1_flt2_dma
47	mdf1_flt3_dma
48	spdifrx_dt_dma
49	spdifrx_cs_dma
50	sai1_a_dma
51	sai1_b_dma
52	sai2_a_dma
53	sai2_b_dma
54	sai3_a_dma
55	sai3_b_dma
56	sai4_a_dma
57	sai4_b_dma
58	adc1_dma
59	adc2_dma
60	tim1_cc1_dma
61	tim1_cc2_dma
62	tim1_cc3_dma
63	tim1_cc4_dma
64	tim1_up_dma

Table 177. Programmed HPDMA1/2/3 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
65	tim1_trig_dma
66	tim1_com_dma
67	tim2_cc1_dma
68	tim2_cc2_dma
69	tim2_cc3_dma
70	tim2_cc4_dma
71	tim2_up_dma
72	tim3_cc1_dma
73	tim3_cc2_dma
74	tim3_cc3_dma
75	tim3_cc4_dma
76	tim3_up_dma
77	tim3_trig_dma
78	tim4_cc1_dma
79	tim4_cc2_dma
80	tim4_cc3_dma
81	tim4_up_dma
82	tim5_cc1_dma
83	tim5_cc2_dma
84	tim5_cc3_dma
85	tim5_cc4_dma
86	tim5_up_dma
87	tim5_trig_dma
88	tim6_up_dma
89	tim7_up_dma
90	tim8_cc1_dma
91	tim8_cc2_dma
92	tim8_cc3_dma
93	tim8_cc4_dma
94	tim8_up_dma
95	tim8_trig_dma
96	tim8_com_dma
97	tim15_cc1_dma
98	tim15_up_dma
99	tim15_trig_dma

Table 177. Programmed HPDMA1/2/3 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
100	tim15_com_dma
101	tim16_cc1_dma
102	tim16_up_dma
103	tim17_cc1_dma
104	tim17_up_dma
105	dcmi_dma
106	Reserved
107	Reserved
108	cryp2_in_dma
109	cryp2_out_dma
110	tim4_cc4_dma
111	lptim1_ic1_dma
112	lptim1_ic2_dma
113	lptim1_ue_dma
114	lptim2_ic1_dma
115	lptim2_ic2_dma
116	lptim2_ue_dma
117	tim2_trig_dma
118	tim4_trig_dma
119	tim15_cc2_dma
120	tim16_com_dma
121	tim17_com_dma
122	i3c1_rx_dma
123	i3c1_tx_dma
124	i3c1_tc_dma
125	i3c1_rs_dma
126	i3c2_rx_dma
127	i3c2_tx_dma
128	i3c2_tc_dma
129	i3c2_rs_dma
130	i3c3_rx_dma
131	i3c3_tx_dma
132	i3c3_tc_dma
133	i3c3_rs_dma
134	lpuart1_rx_dma

Table 177. Programmed HPDMA1/2/3 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
135	lpuart1_tx_dma
136	lptim3_ic1_dma
137	lptim3_ic2_dma
138	lptim3_ue_dma
139	lptim4_ic1_dma
140	lptim4_ic2_dma
141	lptim4_ue_dma
142	hash2_dma

25.3.5 HPDMA block requests

Some HPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in HPDMA_CxTR2 must be set for a correct HPDMA execution of the requested peripheral transfer at the hardware level.

Table 178. Programmed HPDMA1/2/3 request as a block request

HPDMA block requests
lptim1/2_ue_dma

25.3.6 HPDMA channels with peripheral early termination

An HPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 179. HPDMA1/2/3 channel with peripheral early termination

HPDMA channel x with peripheral early termination
x = 0 to 15

This HPDMA support is activated when the channel x is programmed with HPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the HPDMA has transferred the whole block (see *Section 25.4.15* for more details).

Table 180. Programmed HPDMA1/2/3 request with peripheral early termination

Programmed HPDMA channel x request with peripheral early termination
i3c1_rx_dma
i3c2_rx_dma
i3c3_rx_dma

25.3.7 HPDMA triggers

An HPDMA trigger can be assigned to an HPDMA channel x, via TRIGSEL[6:0] in HPDMA_CXTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 181.Programmed HPDMA1/2/3 trigger

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
0	exti2 line0
1	exti2 line1
2	exti2 line2
3	exti2 ine3
4	exti2 line4
5	exti2 line5
6	exti2 line6
7	exti2 line7
8	exti2 line8
9	exti2 line9
10	exti2 line10
11	exti2 line11
12	exti2 line12
13	exti2 line13
14	exti2 line14
15	exti2 line15
16	lptim1_ch1
17	lptim2_ch1
18	lptim3_ch1
19	lptim4_ch1
20	lptim5_ch1
21	rtc_alra_trg
22	rtc_alrb_trg
23	rtc_wut_trg
24	hpdma1_ch0_tc
25	hpdma1_ch1_tc
26	hpdma1_ch2_tc
27	hpdma1_ch3_tc
28	hpdma1_ch4_tc
29	hpdma1_ch5_tc
30	hpdma1_ch6_tc

Table 181.Programmed HPDMA1/2/3 trigger (continued)

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
31	hpdma1_ch7_tc
32	hpdma1_ch8_tc
33	hpdma1_ch9_tc
34	hpdma1_ch10_tc
35	hpdma1_ch11_tc
36	hpdma1_ch12_tc
37	hpdma1_ch13_tc
38	hpdma1_ch14_tc
39	hpdma1_ch15_tc
40	hpdma2_ch0_tc
41	hpdma2_ch1_tc
42	hpdma2_ch2_tc
43	hpdma2_ch3_tc
44	hpdma2_ch4_tc
45	hpdma2_ch5_tc
46	hpdma2_ch6_tc
47	hpdma2_ch7_tc
48	hpdma2_ch8_tc
49	hpdma2_ch9_tc
50	hpdma2_ch10_tc
51	hpdma2_ch11_tc
52	hpdma2_ch12_tc
53	hpdma2_ch13_tc
54	hpdma2_ch14_tc
55	hpdma2_ch15_tc
56	hpdma3_ch0_tc
57	hpdma3_ch1_tc
58	hpdma3_ch2_tc
59	hpdma3_ch3_tc
60	hpdma3_ch4_tc
61	hpdma3_ch5_tc
62	hpdma3_ch6_tc
63	hpdma3_ch7_tc
64	hpdma3_ch8_tc
65	hpdma3_ch9_tc

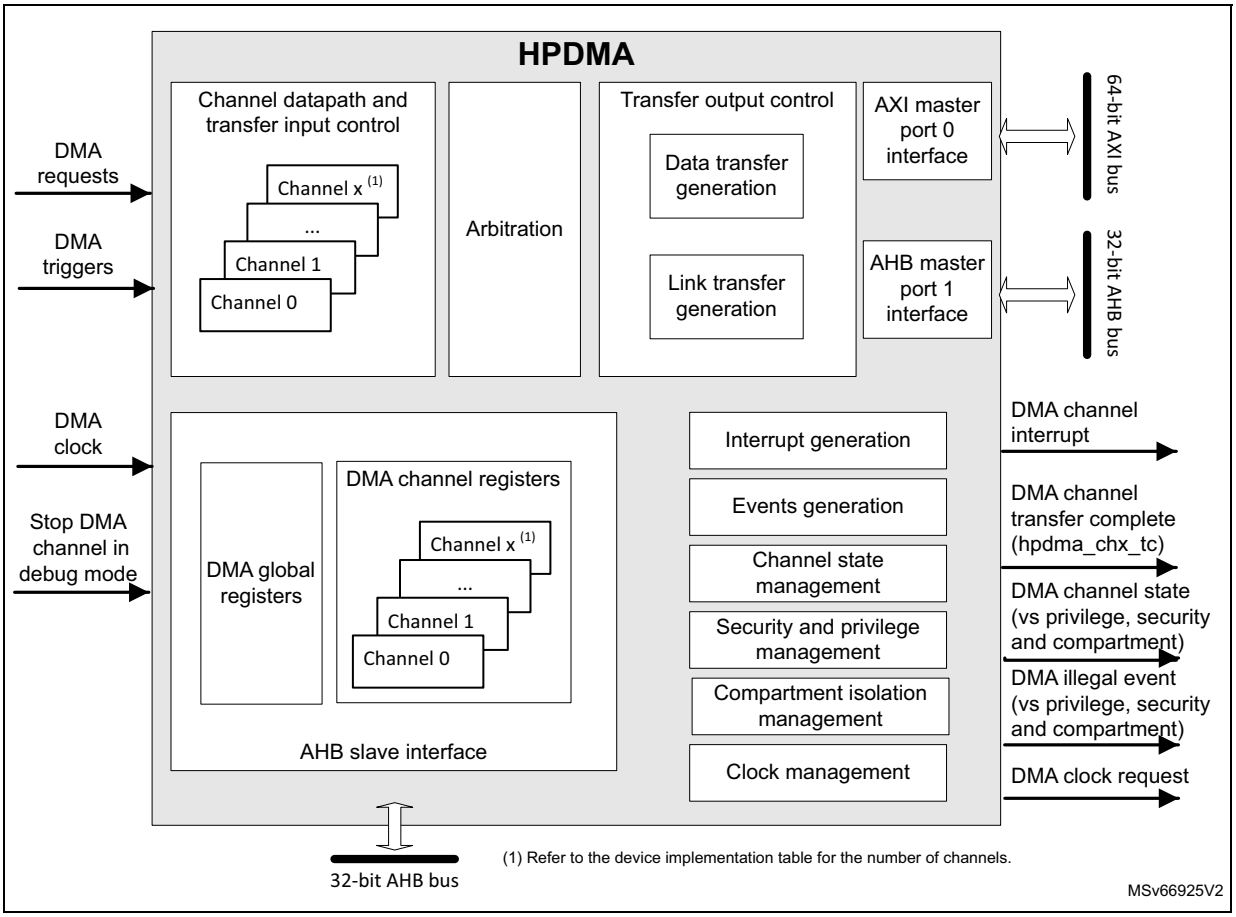
Table 181.Programmed HPDMA1/2/3 trigger (continued)

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
66	hpdma3_ch10_tc
67	hpdma3_ch11_tc
68	hpdma3_ch12_tc
69	hpdma3_ch13_tc
70	hpdma3_ch14_tc
71	hpdma3_ch15_tc
72	tim2_trgo
73	tim15_trgo
74	dcmipp_p2_hsync_evt
75	dcmipp_p2_vsync_evt
76	dcmipp_p2_lineend_evt
77	dcmipp_p2_frameend_evt

25.4 HPDMA functional description

25.4.1 HPDMA block diagram

Figure 135. HPDMA block diagram



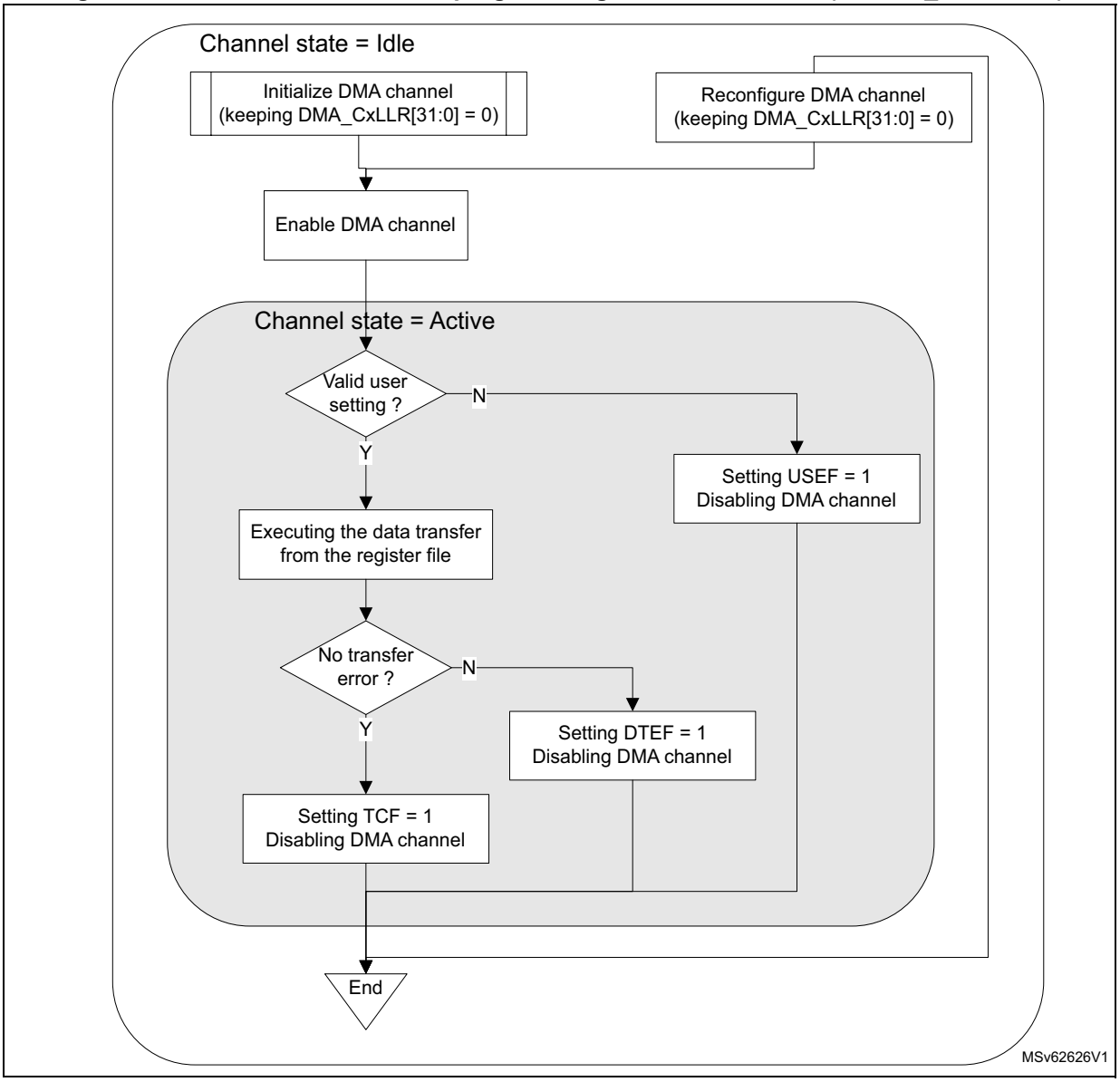
25.4.2 HPDMA channel state and direct programming without any linked-list

After an HPDMA reset, an HPDMA channel x is in idle state. When the software writes 1 into the HPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (HPDMA_CxXXX), switches to the active/non-idle state, and starts to execute the corresponding requested data transfers.

After enabling/starting an HPDMA channel transfer by writing 1 into HPDMA_CxCR.EN, an HPDMA channel interrupt on a complete transfer notifies the software that the HPDMA channel is back in idle state (EN is then deasserted by hardware), and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this HPDMA direct programming without any linked-list (HPDMA_CxLLR = 0).

Figure 136. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0)



25.4.3 HPDMA channel suspend and resume

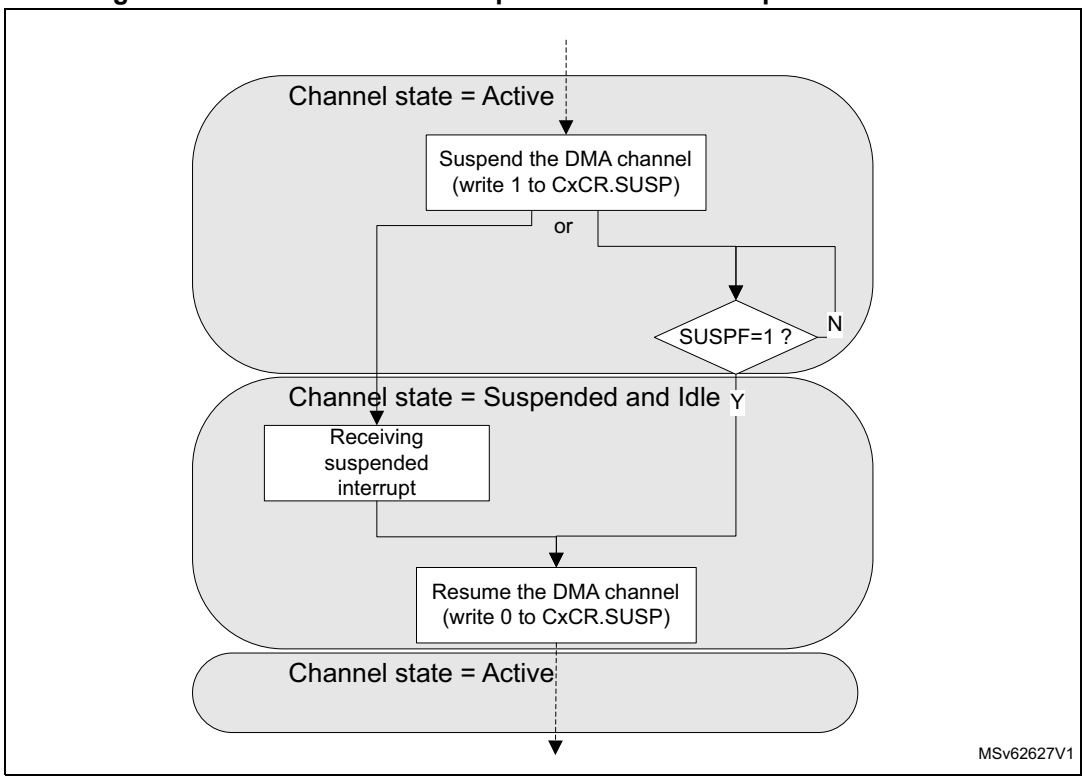
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or bitfield that is hardware modifiable.

Note: An ongoing HPDMA transfer can be a data transfer (a source/destination burst transfer,) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to HPDMACxCR.SUSP.

Figure 137. HPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the HPDMA_*CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.*

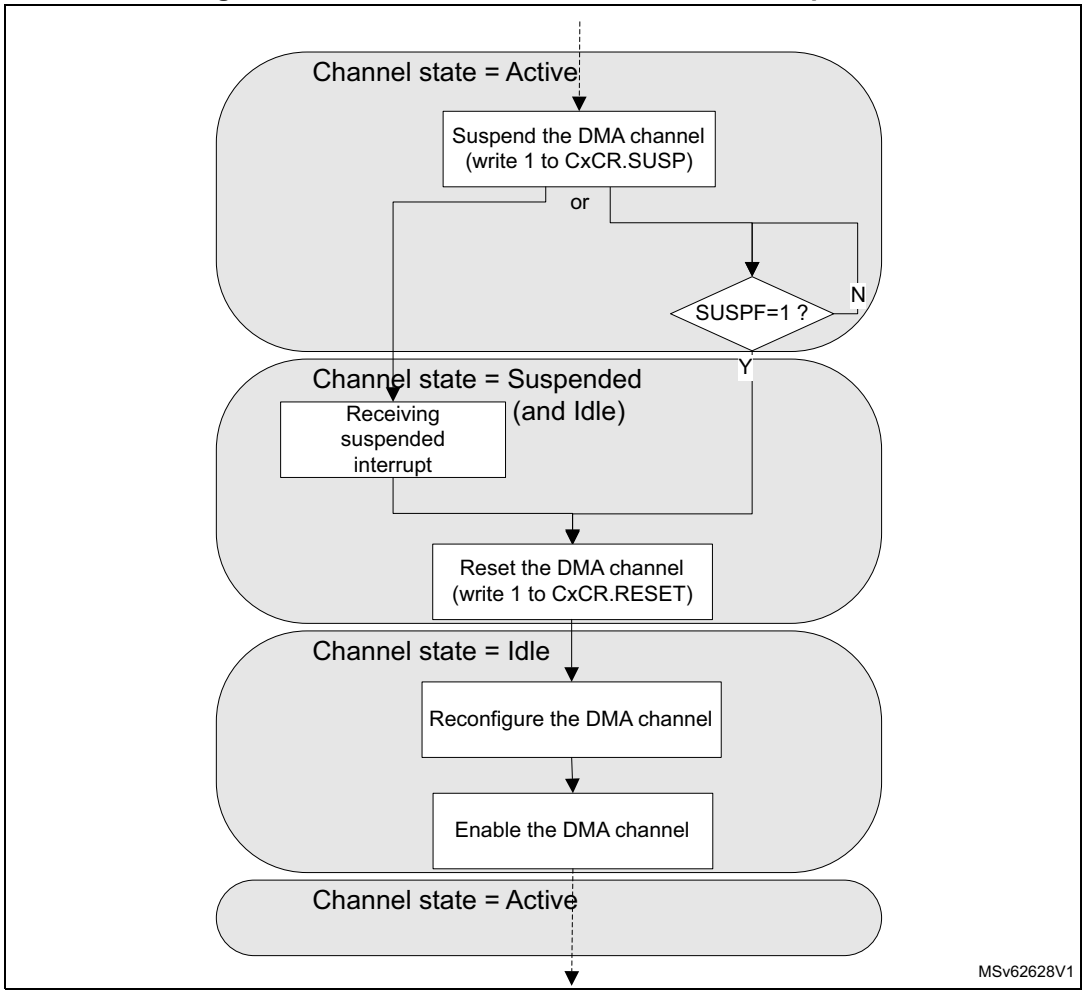
25.4.4 HPDMA channel abort and restart

Alternatively, like for aborting a continuous HPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software resets the channel by writing 1 to HPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the HPDMA_CxCR.EN bit, and the reset of the HPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram hardware-modified HPDMA_CxBR1, HPDMA_CxSAR, and HPDMA_CxDAR.

5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the HPDMA_CxCR.EN bit.

Figure 138. HPDMA channel abort and restart sequence



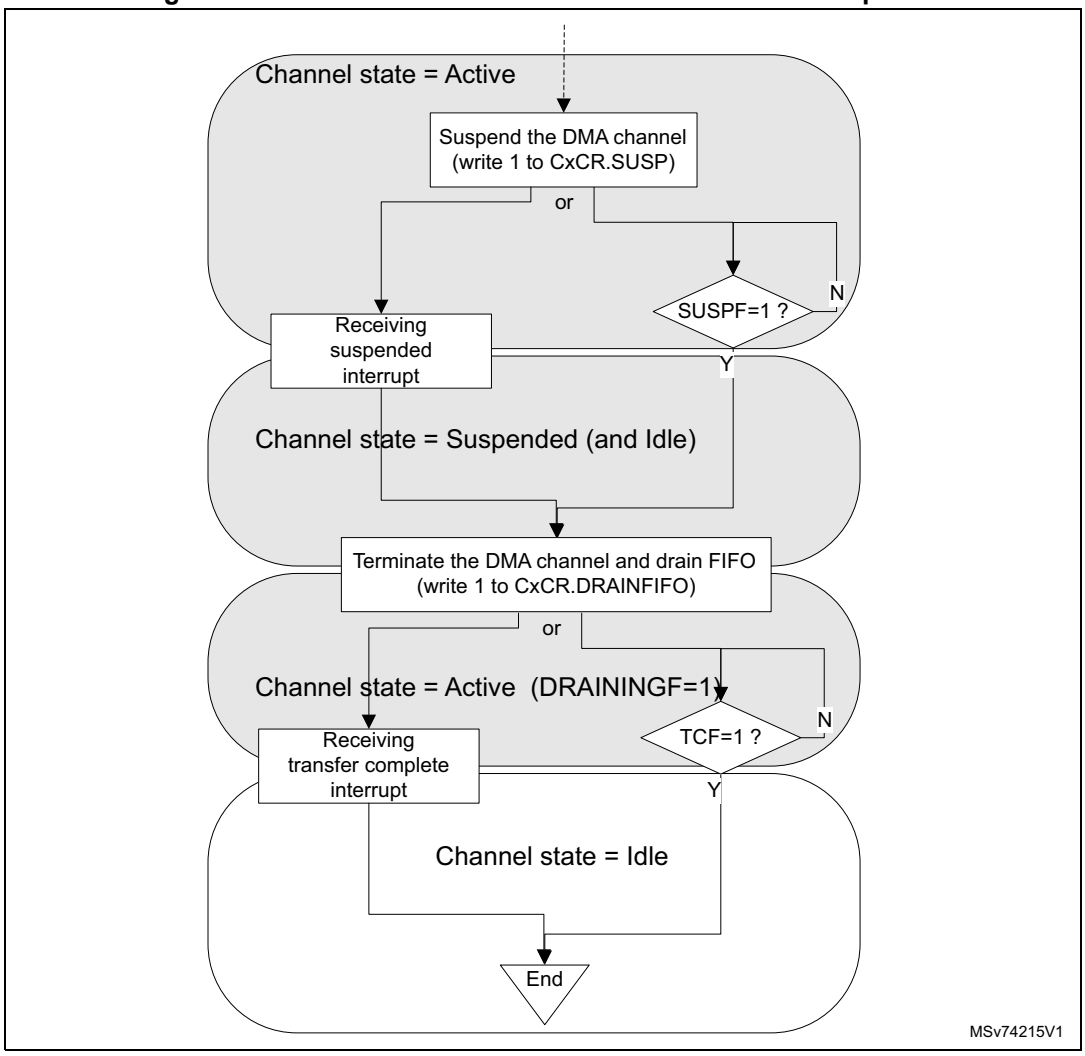
25.4.5 HPDMA channel terminate and drain FIFO

Alternatively, for terminating a continuous HPDMA transfer with a circular buffering or a double buffering, and having popped out any residual byte(s) in the channel FIFO and writing them to the destination, the software can terminate an active channel and drain the channel FIFO with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software orders to drain the channel FIFO by writing 1 to HPDMA_CxCR.DRAINFIFO.
4. The software polls transfer complete flag HPDMA_CxSR.TCF until TCF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.TCIE. Wait for the

channel to be effectively with a drained FIFO means wait for completion of any residual write burst from the FIFO to the destination. Then the channel is disabled by hardware and in IDLE state.

Figure 139. HPDMA channel terminate and drain FIFO sequence



Note: If the softwares writes the HPDMA_CxCR register in a single shot with concurrently SUSP = 1, DRAINFIFO = 1, and RESET = 0, then the sequence for terminating and draining the FIFO also occurs.

HPDMA_CxTR2.DREQ = 1 and HPDMA_CxCR.DRAINFIFO = 1 is not supported, only peripheral-to-memory and memory-to-memory transfers are supported to be terminated with drained FIFO.

25.4.6 HPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

For a channel x, the base address in memory of the data structure of a next LLI_{n+1} is the sum of the following:

- the link base address of the channel x (in HPDMA_CxLBAR)
- the link address offset LA[15:2] bitfield in HPDMA_CxLLR, that is the updated result from the data structure of the previous LLIn of the channel x

The data structure for each LLI can be specific.

A linked-list data structure is addressed following the value of UT1, UT2, UB1, USA, UDA, and ULL bits, plus UB2 and UT3 when present, in HPDMA_CxLLR.

In linked-list mode, each HPDMA linked-list register (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR or HPDMA_CxLLR, plus HPDMA_CxTR3 or HPDMA_CxBR2 when present) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of HPDMA_CxLLR that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present) in HPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x, x = 0 to 11, six contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR (see *Figure 140*) and including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory
- channel x, x = 12 to 15, eight contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 (see *Figure 141*), including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory

Figure 140. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

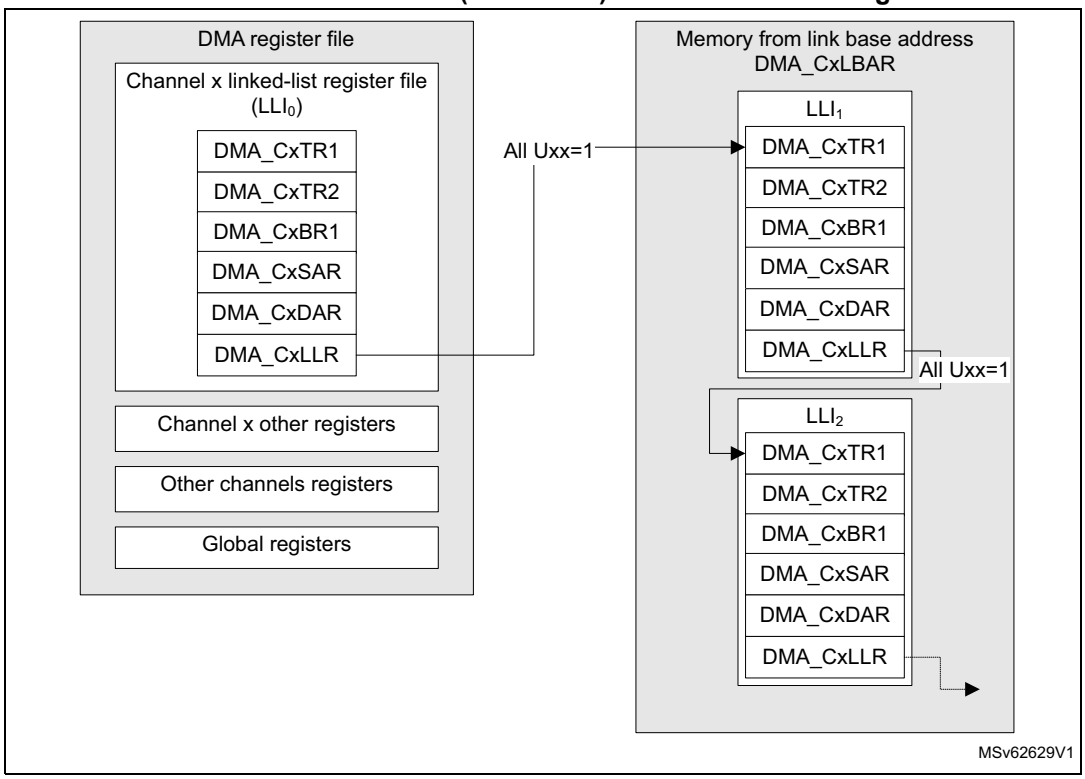
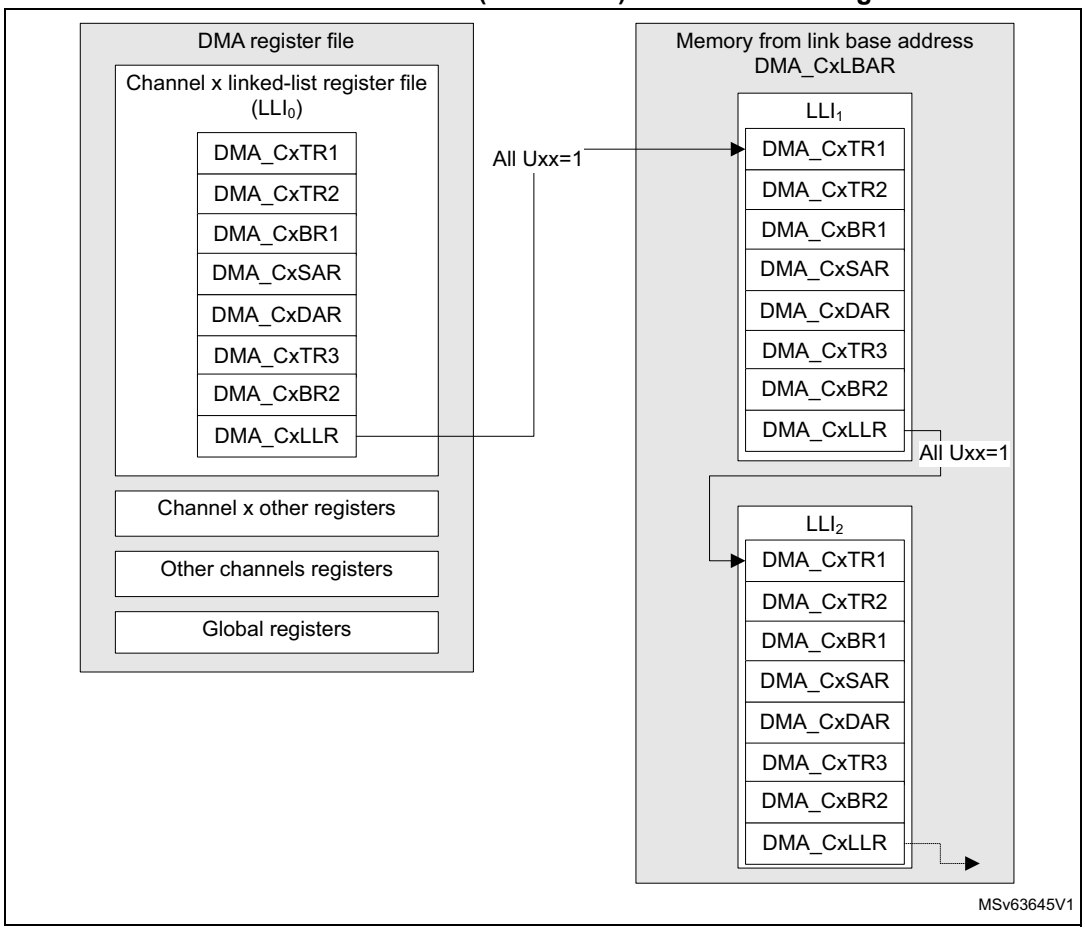


Figure 141. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of HPDMA_CxLLR points to the updated value of HPDMA_CxTR2, instead of HPDMA_CxTR1 which is not to be modified (see *Figure 142*).

Example: if UT1 = UB1 = USA = 0, and if UT3 = UB2 = 0 when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for HPDMA_CxTR1, nor HPDMA_CxBR1, nor HPDMA_CxSAR, nor HPDMA_CxTR3, nor HPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for HPDMA_CxTR2, HPDMA_CxDAR, and HPDMA_CxLLR, as shown in *Figure 143*.

Figure 142. HPDMA dynamic linked-list data structure of linear addressing channel x

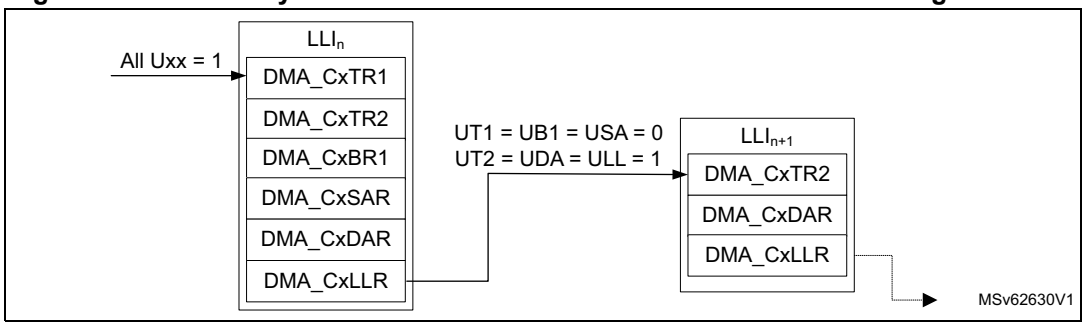
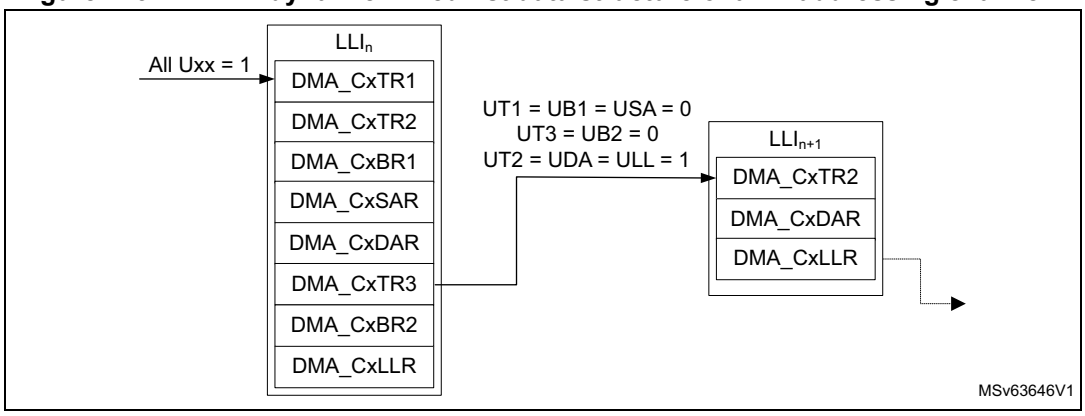


Figure 143. HPDMA dynamic linked-list data structure of a 2D addressing channel x



The user must program HPDMA_CxLLR for each LLI$_n$ to be 32-bit aligned, and not to exceed the 64-Kbyte addressable space pointed by HPDMA_CxLBAR.

25.4.7 Linked-list item transfer execution

An LLI$_n$ transfer is the sequence of:

1. a data transfer: the HPDMA executes the data transfer as described by the HPDMA internal register file (this data transfer can be void/null for LLI$_0$).
2. a conditional link transfer: the HPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI$_{n+1}$, as defined by the HPDMA_CxLLR value of the LLI$_n$.

Note: The initial data transfer, as defined by the internal register file (LLI$_0$), can be null (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxTR2.PFREQ = 0), provided that UB1 is set in HPDMA_CxLLR (meaning there is a non-null data transfer described by the next LLI$_1$ in the memory to be executed).

Depending on the intended HPDMA use, an HPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, HPDMA_CxCR.LSM = 0), or can be programmed for a single execution of a LLI (link step mode, HPDMA_CxCR.LSM = 1), as described in the next sections.

25.4.8 HPDMA channel state and linked-list programming in run-to-completion mode

When HPDMA_CxCR.LSM = 0 (in full-list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), an HPDMA channel x is initially programmed, started by writing 1 to HPDMA_CxCR.EN, and after completed at channel level.
The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present
  - the last LLI_N described by the linked-list data structure in memory, as defined by HPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when HPDMA_CxLLR[31:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if BRC[10:0] is present and HPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

HPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion, and means the following:

- The 16 low significant bits HPDMA_CxLLR.LA[15:0] of the next link address are null.
- All bits HPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present).

The channel may never be completed when HPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI, in one of the following:
  - HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value.
  - HPDMA_CxLLR.ULL = 0
- If LLI_N points to a previous LLI

In the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 25.4.15* for more details).

In the typical run-to-completion mode, the allocation of an HPDMA channel, including its fine programming, is done once during the HPDMA initialization. In order to have a reserved data communication link and HPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

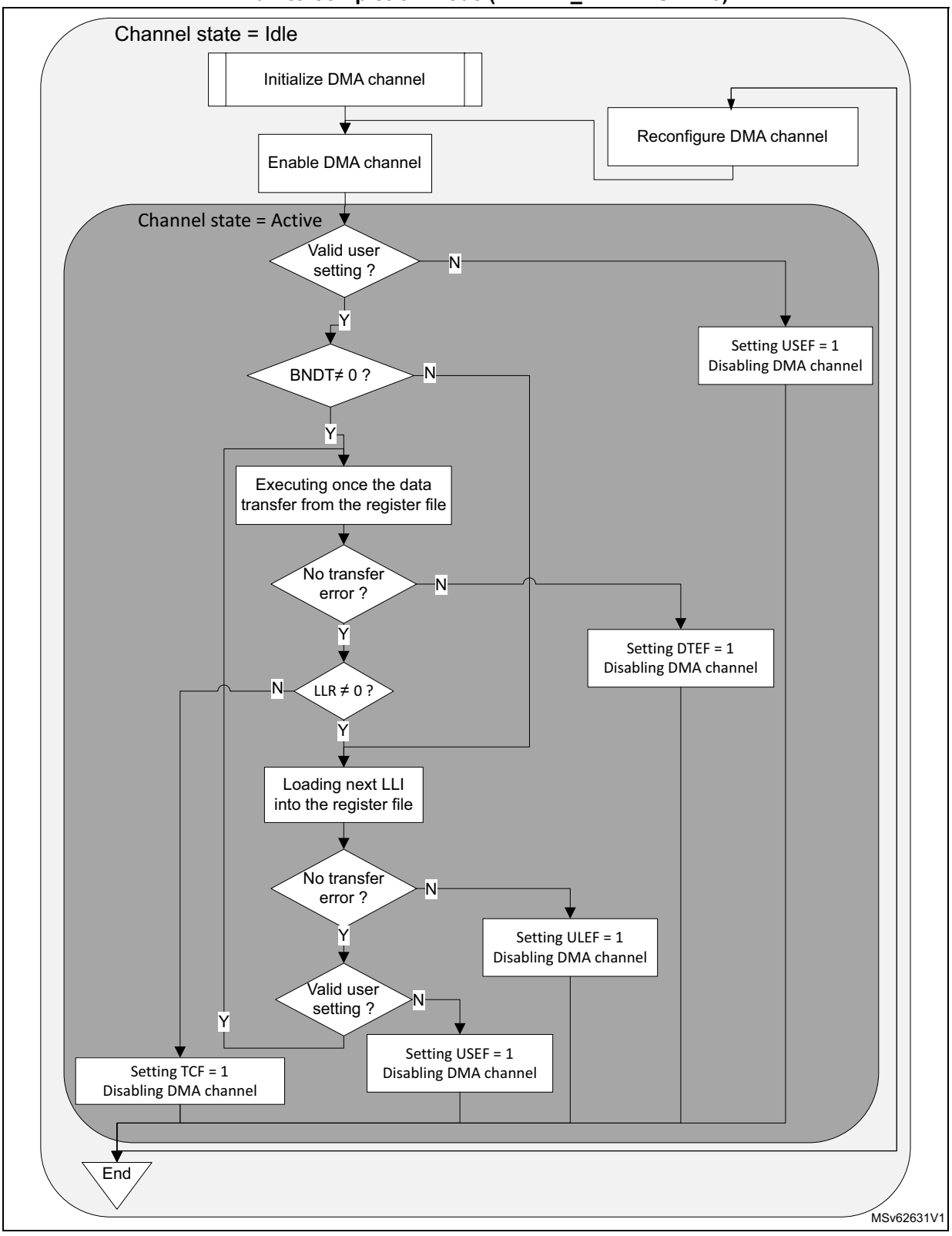
Figure 144 depicts the HPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 144 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel

completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 144, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 25.3.6).

Figure 144. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following bitfields of HPDMA_CxTR2 in the data structure of the LLI_{n-1}:

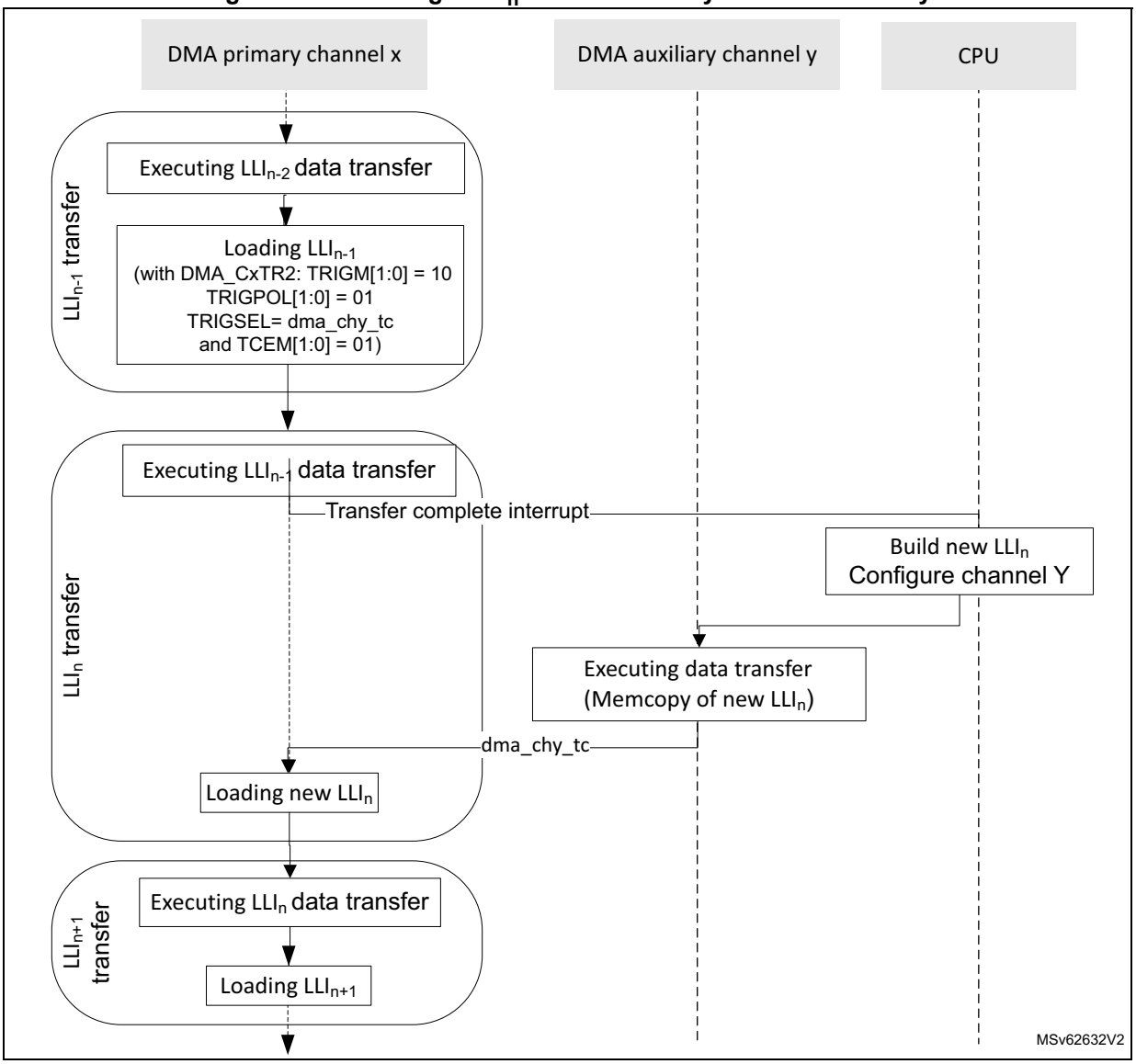
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[6:0] (see *Section 25.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory, and to generate a transfer complete event hpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 145 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with an LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 145. Inserting a LLI_n with an auxiliary HPDMA channel y



25.4.9 HPDMA channel state and linked-list programming in link step mode

When HPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

An HPDMA channel transfer can be programmed at an LLI level, started by writing 1 into HPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with the following:
  - HPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - HPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]), for channel x = 12 to 15, the number of blocks at a 2D/repeated block level (BRC[10:0]+1), and the incrementing/decrementing mode for address offsets.

  - HPDMA_CxTR2 defines the input control (request, trigger), and the output control (transfer-complete event) of the transfer.
  - HPDMA_CxSAR and HPDMA_CxDAR define the source/destination transfer start address.
  - HPDMA_CxTR3 (for channel x = 12 to 15) defines the source/destination additional address offset between burst transfers.
  - HPDMA_CxBR2 (for channel x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - HPDMA_CxLLR defines the data structure, and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when HPDMA_CxBR1.BRC[10:0] = 0 if BRC[10:0] is present, and HPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the HPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If an LLI is recursive (pointing to itself as a next LLI, either HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value, or HPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer can be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 25.4.15* for more details).

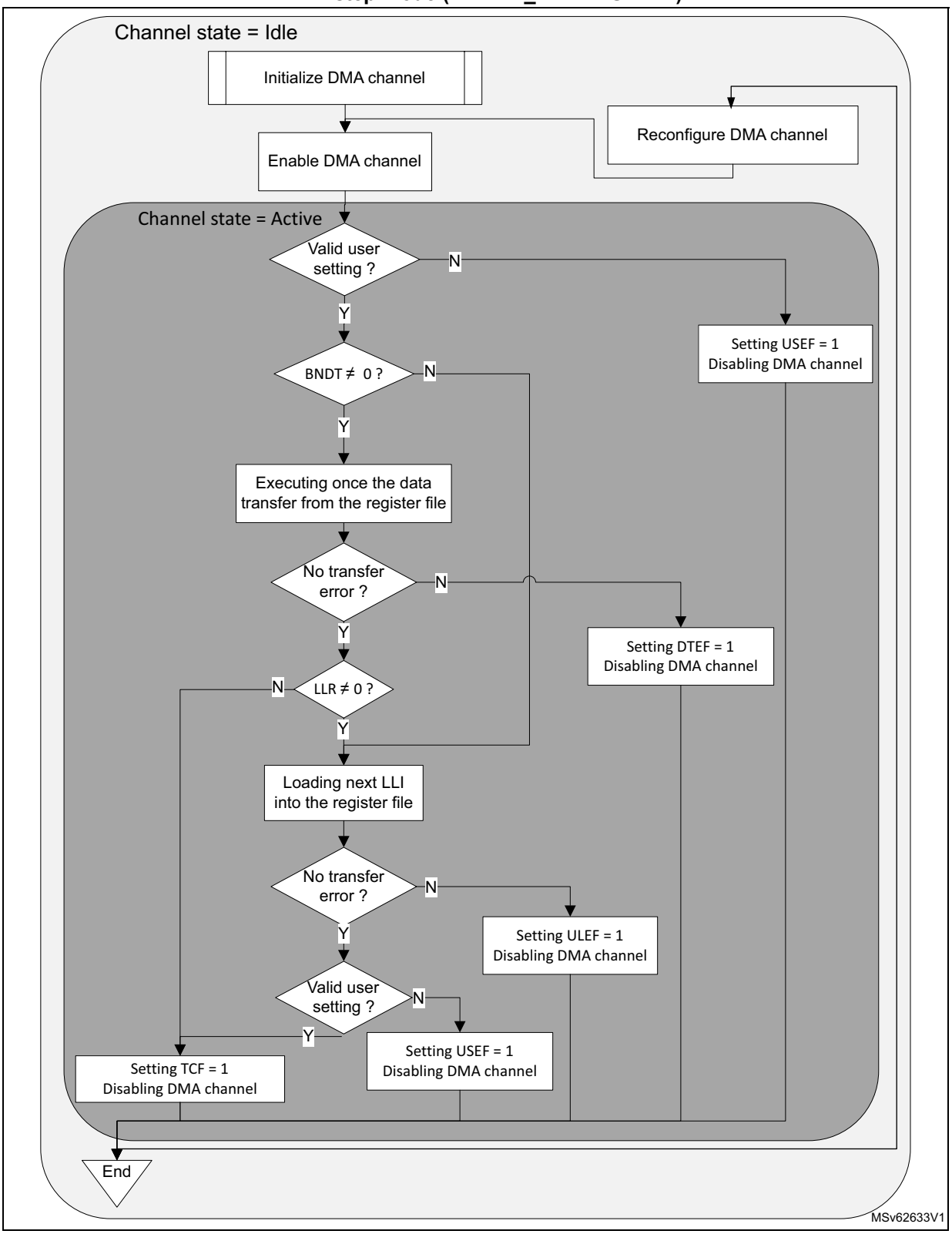
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of HPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 146 depicts the HPDMA channel execution mode, and its programming in link step mode.

Note: Figure 146 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 146, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 25.3.6).

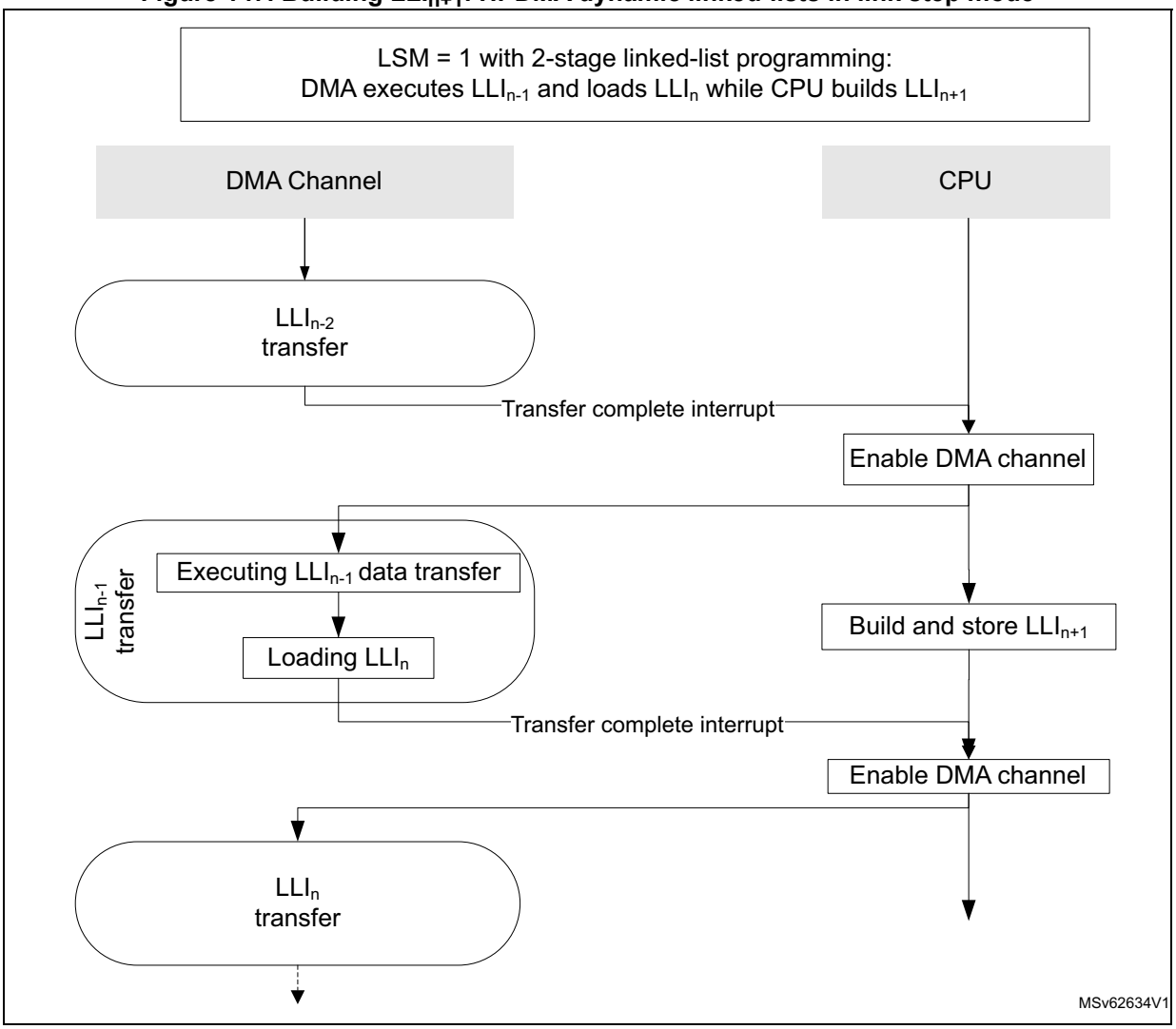
Figure 146. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after the HPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

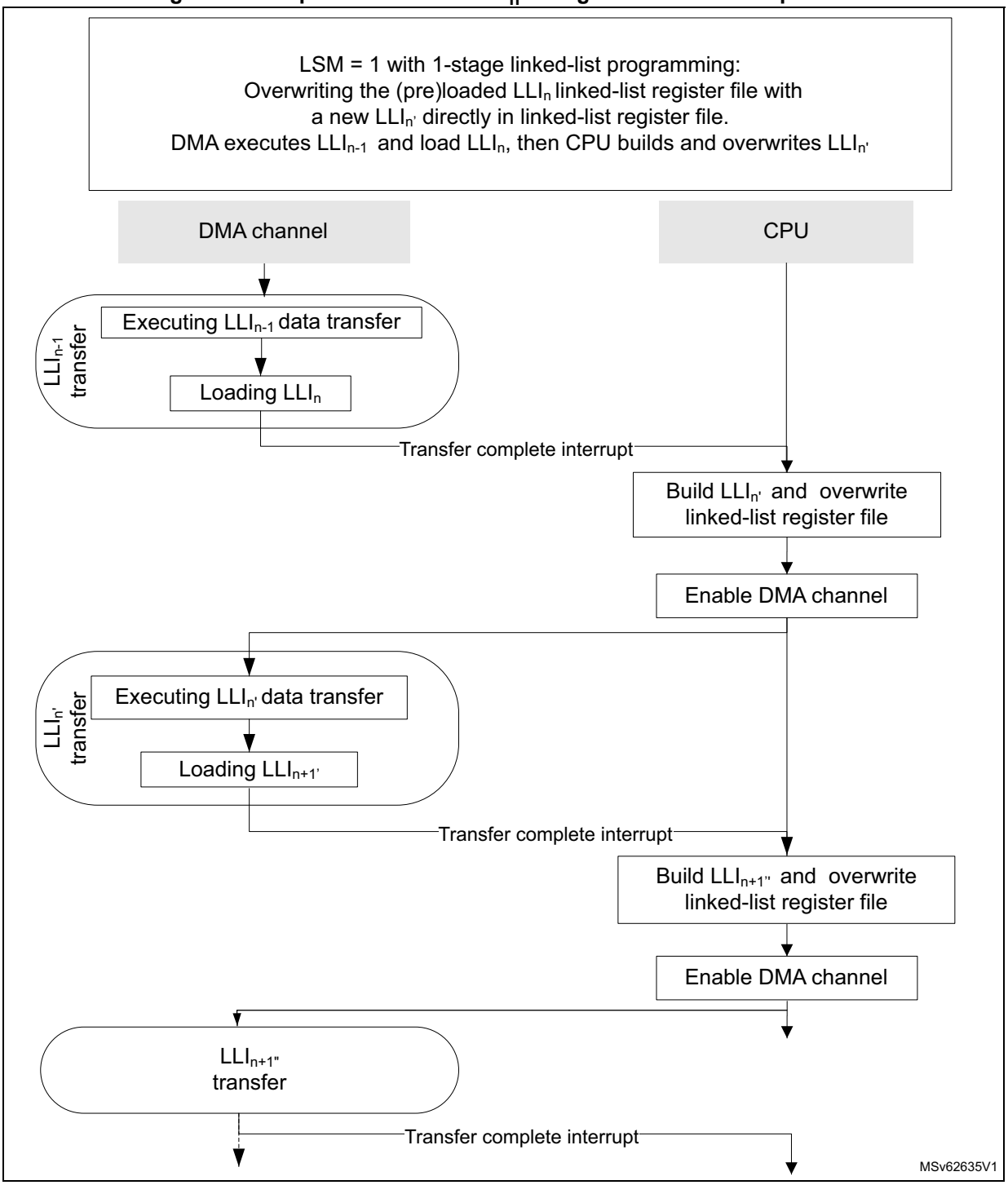
Figure 147. Building LLI_{n+1}: HPDMA dynamic linked-lists in link step mode



Run-time replacing an LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after the HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 148*.

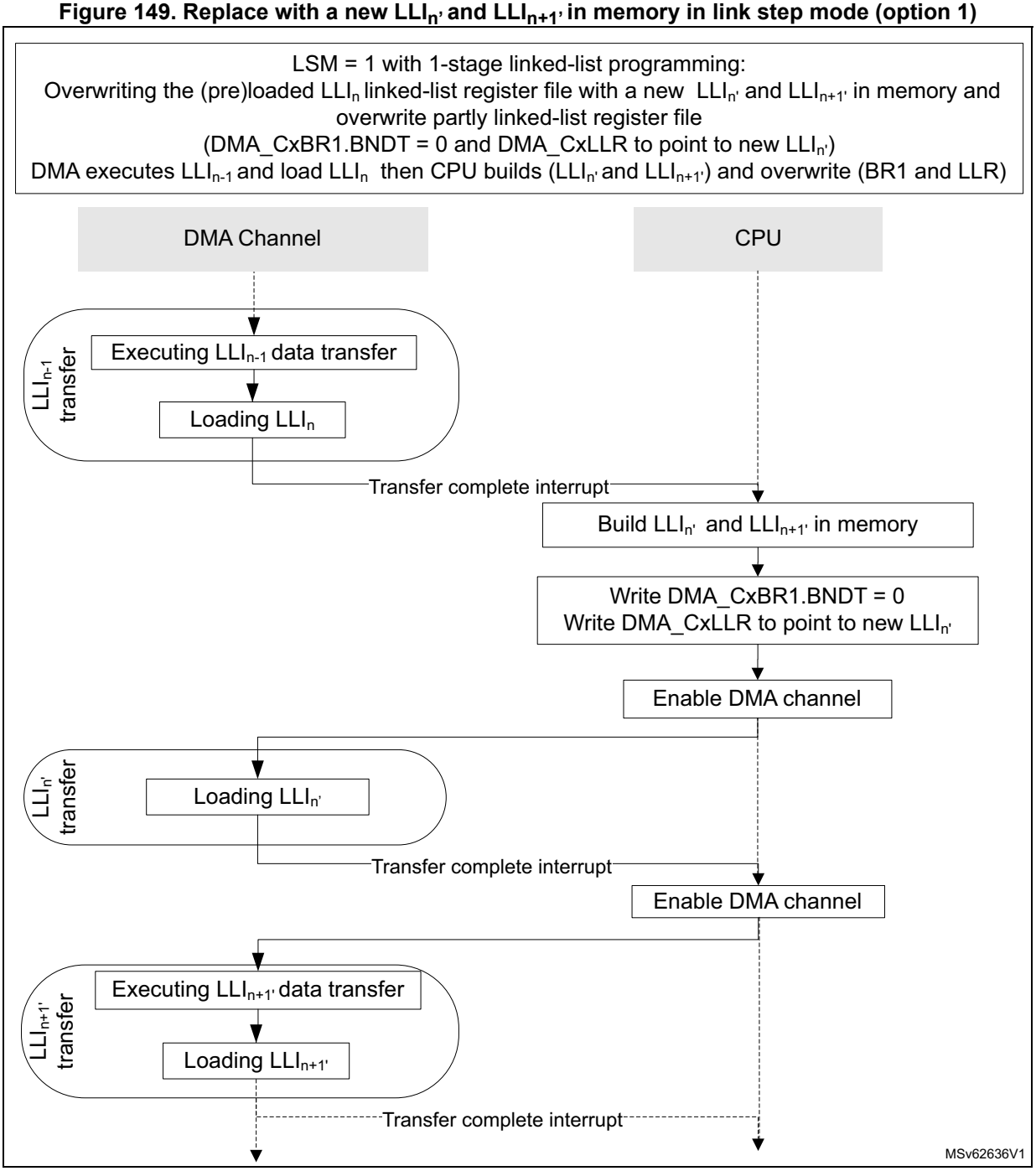
Figure 148. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing an LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (HPDMA_CxBR1.BNDT[15:0] to be null, and HPDMA_CxLLR to point to new LLI_n) as shown in *Figure 149*.

Figure 149. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



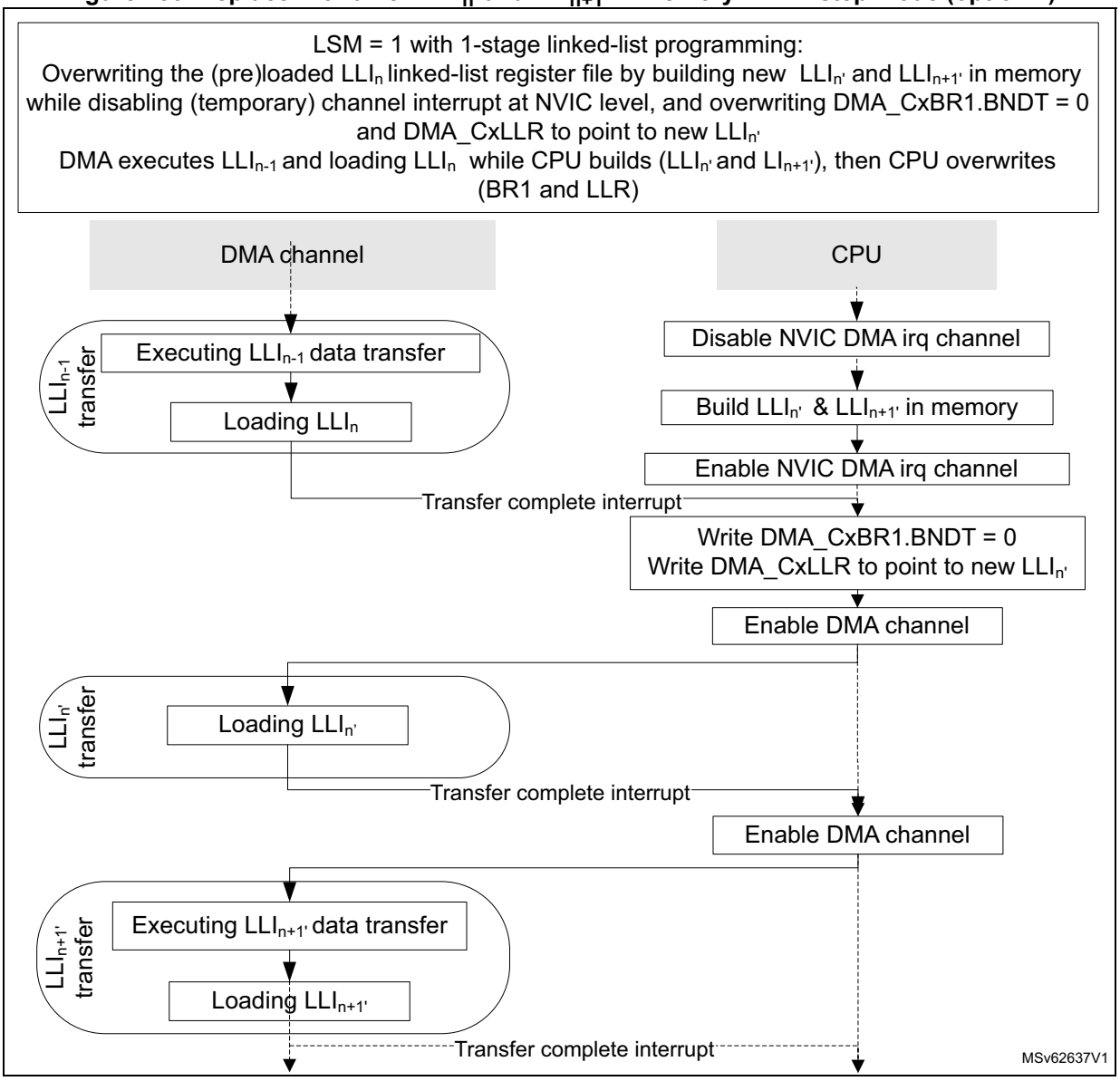
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile the HPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite HPDMA_CxBR1.BNDT[15:0] to be null and HPDMA_CxLLR to point to new $LLI_{n'}$, as shown in the figure below.

Figure 150. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



25.4.10 HPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (HPDMA_CxCR.EN = 0), and update the execution mode (HPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the HPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the HPDMA channel
- abort and reconfigure the channel with an LSM update (see *Section 25.4.4*.)

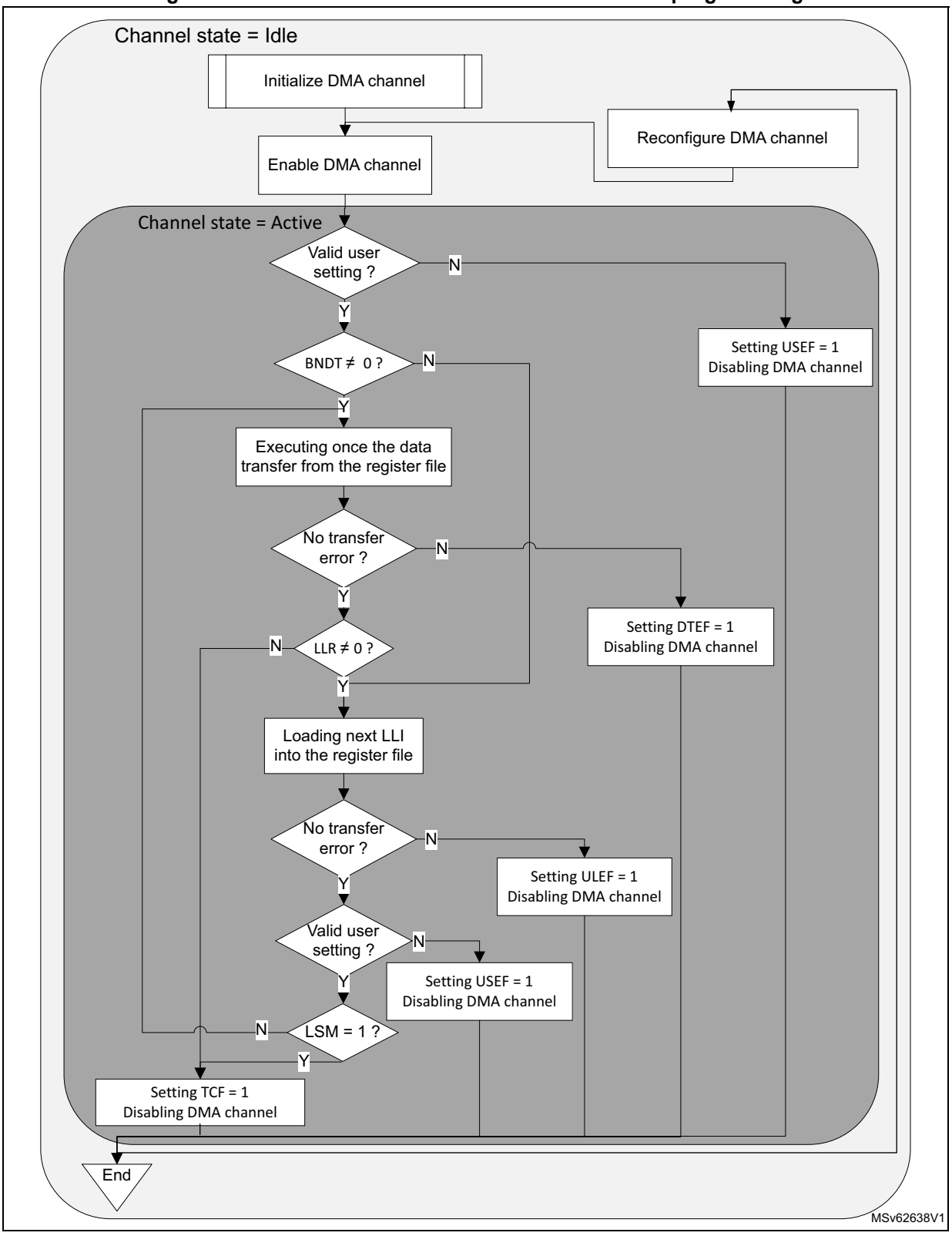
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 151 shows the overall and unified HPDMA linked-list programming, whatever is the execution mode.

Note: Figure 151 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 151, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 25.3.6).

Figure 151. HPDMA channel execution and linked-list programming



25.4.11 HPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 25.3.1* for more details).

HPDMA burst

A programmed transfer at the lowest level is an HPDMA burst.

An HPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in HPDMA_CxTR1.

The addressing mode after each data (named beat) of an HPDMA burst is defined by SINC and DINC in HPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of an HPDMA source/destination burst (defined by HPDMA_CxSAR and HPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of an HPDMA burst.

Table 182. Programmed HPDMA source/destination burst

SAP/DAP (allocated port)	SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/ beats)	Next data/ beat address	Next burst address	Burst address alignment
0: AXI 1: AHB	00	1	0 (fixed)	n = 0 to 63(1)(2)	n+1	+ 0	+ 0	1
	01	2						2
	10	4						4
0: AXI	11	8						8
0: AXI 1: AHB	00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
	01	2				+ 2	+ 2 * (n + 1)	2
	10	4				+ 4	+ 4 * (n + 1)	4
0: AXI	11	8				+ 8	+ 8 * (n + 1)	8
0: AHB	11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.
2. As highlighted in *Section 25.3.2*, the maximum allowed AXI burst length is 16. The user must set SBL_1[5:0] lower or equal to 15 if the source allocated port is AXI (if SAP = 0). The user must set DBL_1[5:0] lower or equal to 15 if the destination allocated port is AXI (if DAP = 0).

The next burst address in the above table is the next source/destination default address pointed by HPDMA_CxSAR or HPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

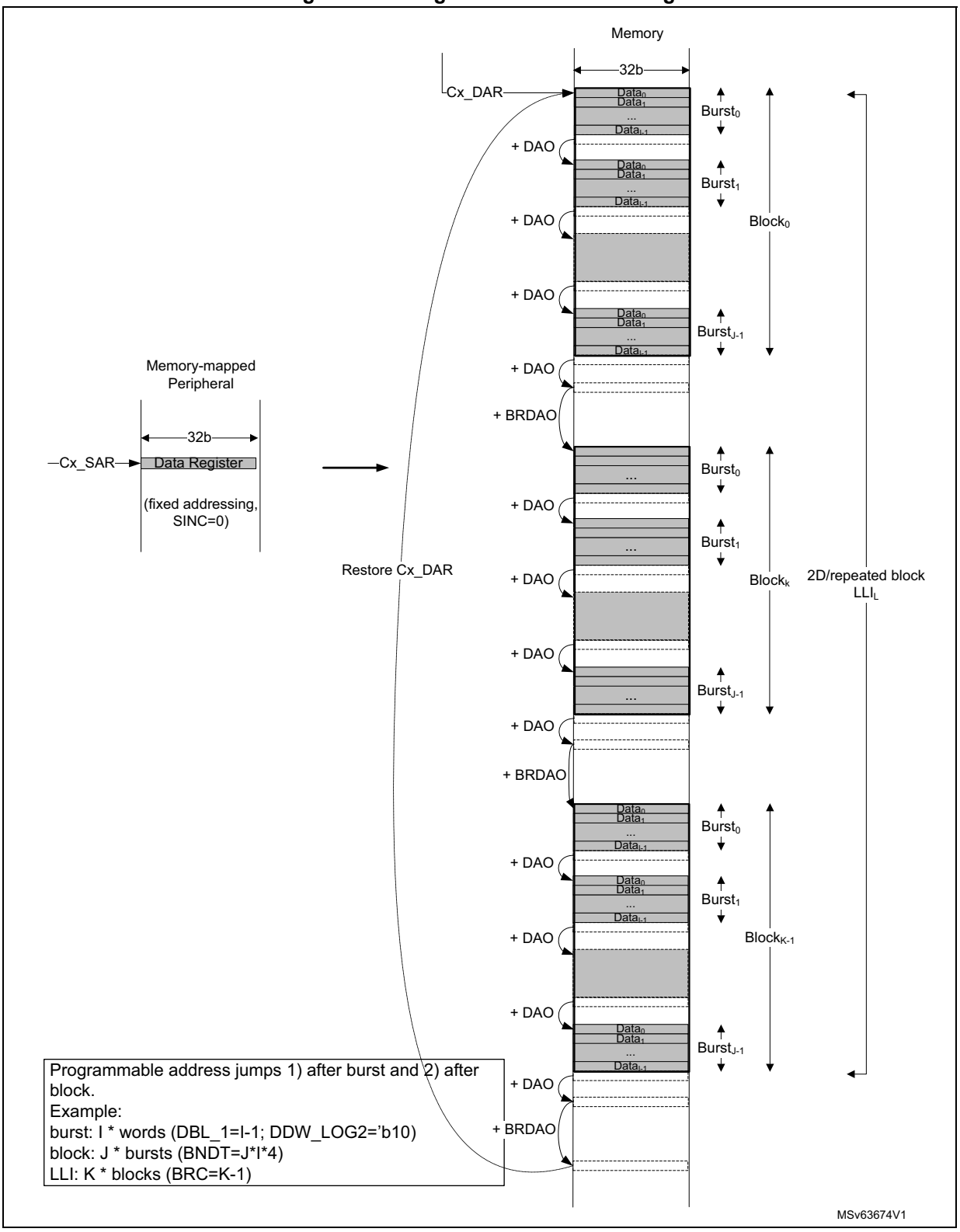
HPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by HPDMA_CxTR2.SAO[12:0]/DAO[12:0] and HPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by HPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and HPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 152* with a 2D destination buffer.

Figure 152. Programmed 2D addressing



HPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by HPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by HPDMA_CxTR1.DAP

HPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half of the FIFO size of the channel x, in bytes (see *Section 25.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (HPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

HPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

HPDMA burst versus source block size, 1- or 4-Kbyte address boundary and FIFO size

The programmed source/destination HPDMA burst is implemented with an AHB/AXI burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination HPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (HPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (HPDMA_CxTR1.SDW_LOG2[1:0]), the HPDMA modifies and shortens bursts into

singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1- or 4-Kbyte address boundary on, respectively, an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, or if the source/destination burst on an AXI transfer is both with fixed addressing and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.

In any case, the HPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in HPDMA_CxSR) is compared to and updated with the effective transfer size, and the HPDMA re-arbitrates between each AHB/AXI single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the HPDMA read transfers related to the update of the linked-list parameters from the memory to the internal HPDMA registers, are scheduled over the link allocated port, as programmed by HPDMA_CxCR.LAP.

HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment

The data handling is controlled by HPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word, word, or double-word, as per SDW_LOG2[1:0] and DDW_LOG2[1:0] (see *Table 183*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if the source data width is a word or a double-word (for AXI source bus, SAP = 0), the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data-width conversion by packing, unpacking, padding, or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.
   - If DWX = 1and if the destination data width is a double-word and the destination bus is AXI (DAP = 0), the two aligned words are exchanged within aligned (post PAM[1:0]) double-words).

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding, and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 183. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)(3)(4)	0			$0B_3,0B_2,0B_1,0B_0$
							1			B_30,B_20,B_10,B_00
						01 (RA, SE)(3)(4)	0			SB_3,SB_2,SB_1,SB_0
							1			B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$000B_1,000B_0$
							1			$00B_10,00B_00$
							0	1		$0B_100,0B_000$
							1			B_1000,B_0000
						01 (RA, SE)(3)(4)	0	0		$SSSB_1,SSSB_0$
							1			SSB_1S,SSB_0S
							0	1		SB_1SS,SB_0SS
							1			B_1SSS,B_0SSS
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream[1]	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0][2]	DBX	DHX	DWX	Destination data stream[1]
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	11[5]	Double-word	00 (RA, 0P)[3][4]	0	0	0	$0000000B_0$
							1			$000000B_00$
							0	1		$00000B_000$
							1			$0000B_0000$
							0	0	1	$000B_00000$
							1			$00B_000000$
							0	1		$0B_0000000$
							1			$B_00000000$
						01 (RA, SE)[3][4]	0	0	0	$SSSSSSSB_0$
							1			$SSSSSSB_0S$
							0	1		$SSSSSB_0SS$
							1			$SSSSB_0SSS$
							0	0	1	$SSSB_0SSSS$
							1			SSB_0SSSSS
							0	1		$SB_0SSSSSS$
							1			$B_0SSSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,$ B_3B_2,B_1B_0	x	00	Byte	00 (RA, LT)(3)	x	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)(3)				B_7,B_5,B_3,B_1
						1x (UNPACK)				$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	xx	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$00B_3B_2,00B_1B_0$
							1			B_3B_200,B_1B_000
							0	1		B_2B_300,B_0B_100
							1			$00B_2B_3,00B_0B_1$
						01 (RA, SE)(3)(4)	0	0		SSB_3B_2,SSB_1B_0
							1			B_3B_2SS,B_1B_0SS
							0	1		B_2B_3SS,B_0B_1SS
							1			SSB_2B_3,SSB_0B_1
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	B_7B_6,B_5B_4, B_3B_2,B_1B_0	x	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$000000B_1B_0$
							1			$000000B_0B_1$
							0	1		$0000B_1B_000$
							1			$0000B_1B_000$
							0	0	1	$00B_1B_00000$
							1			$00B_0B_10000$
							0	1		$B_1B_0000000$
							1			$B_0B_1000000$
						01 (RA, SE)(3)(4)	-	0	0	$SSSSSSB_1B_0$
							1			$SSSSSSB_0B_1$
							0	1		$SSSSB_1B_0SS$
							1			$SSSSB_1B_0SS$
							0	0	1	SSB_1B_0SSSS
							1			SSB_0B_1SSSS
							0	1		$B_1B_0SSSSSS$
							1			$B_0B_1SSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_5B_4,B_1B_0
							1			B_4B_5,B_0B_1
						01 (LA, RT)(3)	0			B_7B_6,B_3B_2
							1			B_6B_7,B_2B_3
						1x (UNPACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	xx	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream[1]	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0][2]	DBX	DHX	DWX	Destination data stream[1]
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	11[5]	Double-word	1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
						00 (RA, 0P)[3][4]	0	0	0	$0000B_3B_2B_1B_0$
							1			$0000B_2B_3B_0B_1$
							0	1		$0000B_1B_0B_3B_2$
							1			$0000B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_00000$
							1			$B_1B_0B_3B_20000$
							0	1		$B_3B_2B_1B_00000$
							1			$B_0B_1B_2B_30000$
						01 (RA, SE)[3][4]	0	0	0	$SSSSB_3B_2B_1B_0$
							1			$SSSSB_2B_3B_0B_1$
							0	1		$SSSSB_1B_0B_3B_2$
							1			$SSSSB_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0SSSS$
							1			$B_1B_0B_3B_2SSSS$
							0	1		$B_3B_2B_1B_0SSSS$
							1			$B_0B_1B_2B_3SSSS$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_6B_4,B_2B_0
							1			B_4B_6,B_0B_2
						01 (LA, RT)(3)	0			B_7B_5,B_3B_1
							1			B_5B_7,B_1B_3
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream[1]	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0][2]	DBX	DHX	DWX	Destination data stream[1]
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	11[5]	Double-word	00 (RA, 0P)[3][4]	0	0	0	$0000B_3B_1B_2B_0$
							1			$0000B_1B_3B_0B_2$
							0	1		$0000B_2B_0B_3B_1$
							1			$0000B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_00000$
							1			$B_1B_3B_0B_20000$
							0	1		$B_2B_0B_3B_10000$
							1			$B_0B_2B_1B_30000$
						01 (RA, SE)[3][4]	0	0	0	$SSSSB_3B_1B_2B_0$
							1			$SSSSB_1B_3B_0B_2$
							0	1		$SSSSB_2B_0B_3B_1$
							1			$SSSSB_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0SSSS$
							1			$B_1B_3B_0B_2SSSS$
							0	1		$B_2B_0B_3B_1SSSS$
							1			$B_0B_2B_1B_3SSSS$
						1x (PACK)	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)	-			B_{31},B_{23},B_{15},B_7
						1x (UNPACK)	x			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_9B_8,B_1B_0
							1			B_8B_9,B_0B_1
						01 (LA, RT)(3)	0			$B_{15}B_{14},B_7B_6$
							1			$B_{14}B_{15},B_6B_7$
						1x (UNPACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, LT)(3)	0	0		$B_{11}B_{10}B_9B_8,$ $B_3B_2B_1B_0$
							1			$B_{10}B_{11}B_8B_9,$ $B_2B_3B_0B_1$
							0	1		$B_9B_8B_{11}B_{10},$ $B_1B_0B_3B_2$
							1			$B_8B_9B_{10}B_{11},$ $B_0B_1B_2B_3$
						01 (LA, RT)(3)	0	0		$B_{15}B_{14}B_{13}B_{12},$ $B_7B_6B_5B_4$
							1			$B_{14}B_{15}B_{12}B_{13},$ $B_6B_7B_4B_5$
							0	1		$B_{13}B_{12}B_{15}B_{14},$ $B_5B_4B_7B_6$
							1			$B_{12}B_{13}B_{14}B_{15},$ $B_4B_5B_6B_7$
						1x (UNPACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	0	11(5)	Double-word	xx	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
			1	00	Byte	00 (RA, LT)(3)	x	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)				B_{31},B_{23},B_{15},B_7
						1x (UNPACK)				$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			$B_{10}B_8,B_2B_0$
							1			B_8B_{10},B_0B_2
						01 (LA, RT)(3)	0			$B_{15}B_{13},B_7B_5$
							1			$B_{13}B_{15},B_5B_7$
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$

Table 183. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	1	10	Word	00 (RA, LT)(3)	0	0	x	$B_{11}B_9B_{10}B_8,B_3B_1B_2B_0$
							1			$B_9B_{11}B_8B_{10},B_1B_3B_0B_2$
							0	1		$B_{10}B_8B_{11}B_9,B_2B_0B_3B_1$
							1			$B_8B_{10}B_9B_{11},B_0B_2B_1B_3$
						01 (LA, RT)(3)	0	0		$B_{15}B_{13}B_{14}B_{12}$, $B_7B_5B_6B_4$
							1			$B_{13}B_{15}B_{12}B_{14}$, $B_5B_7B_4B_6$
							0	1		$B_{14}B_{12}B_{15}B_{13}$, $B_6B_4B_7B_5$
							1			$B_{12}B_{14}B_{13}B_{15}$, $B_4B_6B_5B_7$
						1x (UNPACK)	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$
				11(5)	Double-word	xx	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
3. RA= right aligned. LA = left aligned. RT = right truncated. LT = left truncated.
4. 0P= zero-bit padding up to the destination data width. SE = sign bit extended up to the destination data width.
5. if DDW_LOG2[1:0] = 11 and if DAP = 0 (destination allocated port is AXI). Else if DDW_LOG2[1:0] = 11 and DAP = 1 (AHB), a user setting error (USEF) is reported.
6. if SDW_LOG2[1:0] = 11 and if SAP = 0 (source allocated source port is AXI). Else if SDW_LOG2[1:0] = 11 and SAP = 1 (AHB), a user setting error (USEF) is reported.

25.4.12 HPDMA transfer request and arbitration

HPDMA transfer request

As defined by HPDMA_CxTR2, a programmed HPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the HPDMA hardware peripheral request is driven by REQSEL[7:0] (see *Section 25.3.5*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in HPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control register update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign the same input hardware peripheral HPDMA request via HPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral, and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

HPDMA transfer request for arbitration

A ready FIFO-based HPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (HPDMA_CxTR1.SAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

A ready FIFO-based HPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (HPDMA_CxTR1.DAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

An arbitrated HPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, HPDMA_CxTR3, HPDMA_CxBR2 or HPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (HPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (HPDMA_CxTR2.SWREQ = 1), or with a hardware request (HPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (HPDMA_CxTR2.SWREQ = 0 and HPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request

to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

HPDMA arbitration

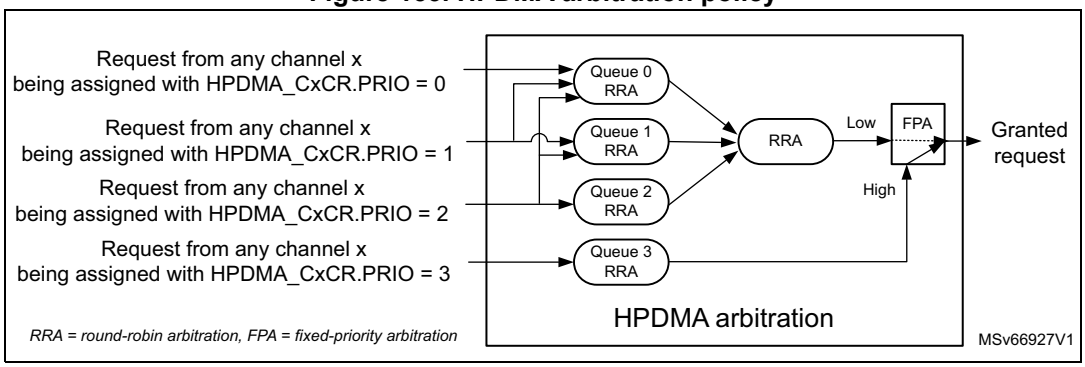
The HPDMA arbitration is directed from the 4-grade assigned channel priority (HPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 153*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 153. HPDMA arbitration policy



HPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- equal maximum bandwidth between requests with same priority
- reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive HPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any HPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one HPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of HPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the HPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

25.4.13 HPDMA triggered transfer

A programmed HPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by HPDMA_CxTR2.TRIGPOL[1:0] and HPDMA_CxTR2.TRIGSEL[6:0] (see *Section 25.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in HPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block (for channel x = 12 to 15), or each programmed burst read/write. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The HPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

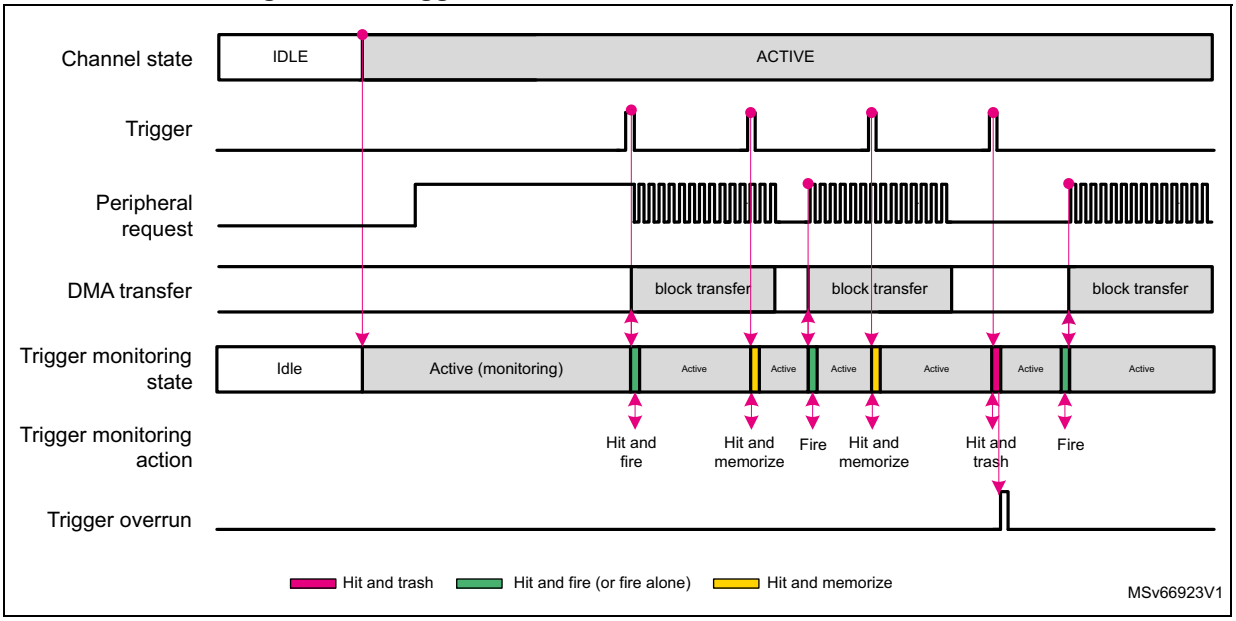
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[6:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates HPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger, hit_{n+1} is memorized. If another trigger hit_{n+2} is detected, and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization, and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 154. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

25.4.14 HPDMA circular buffering with linked-list programming

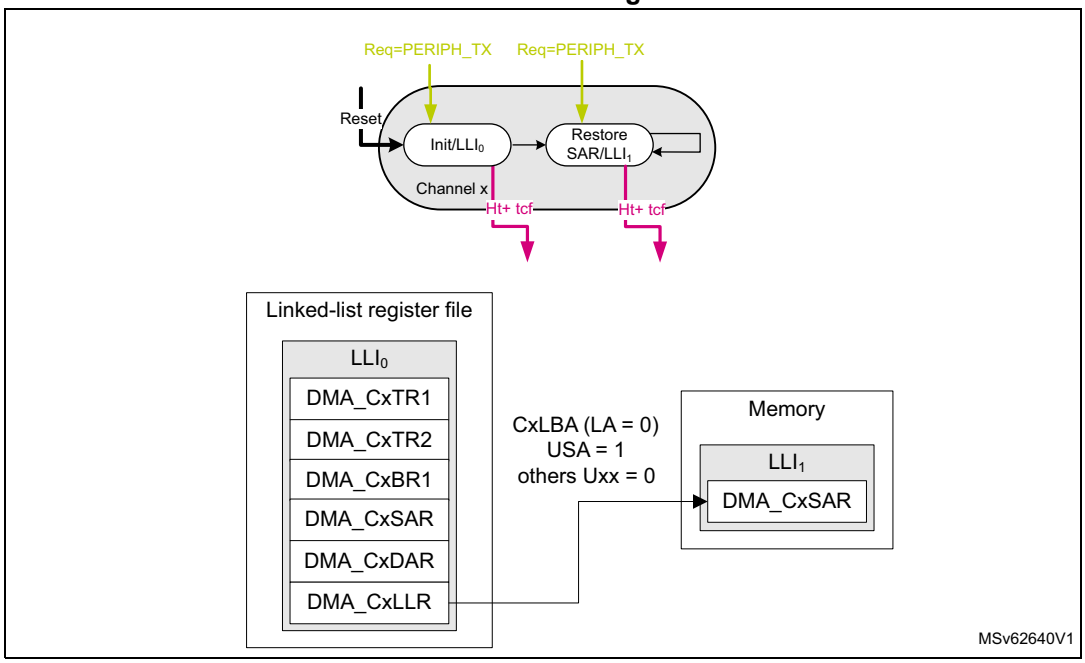
HPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete-transfer event/interrupt generation (HPDMA_CxCR.HTIE = 1 and HPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel.
A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address for the memory-to-peripheral transfer (respectively the peripheral-to-memory).
The HPDMA automatically reloads the initially programmed HPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore HPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing HPDMA channel and a source circular buffer.

Figure 155. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel

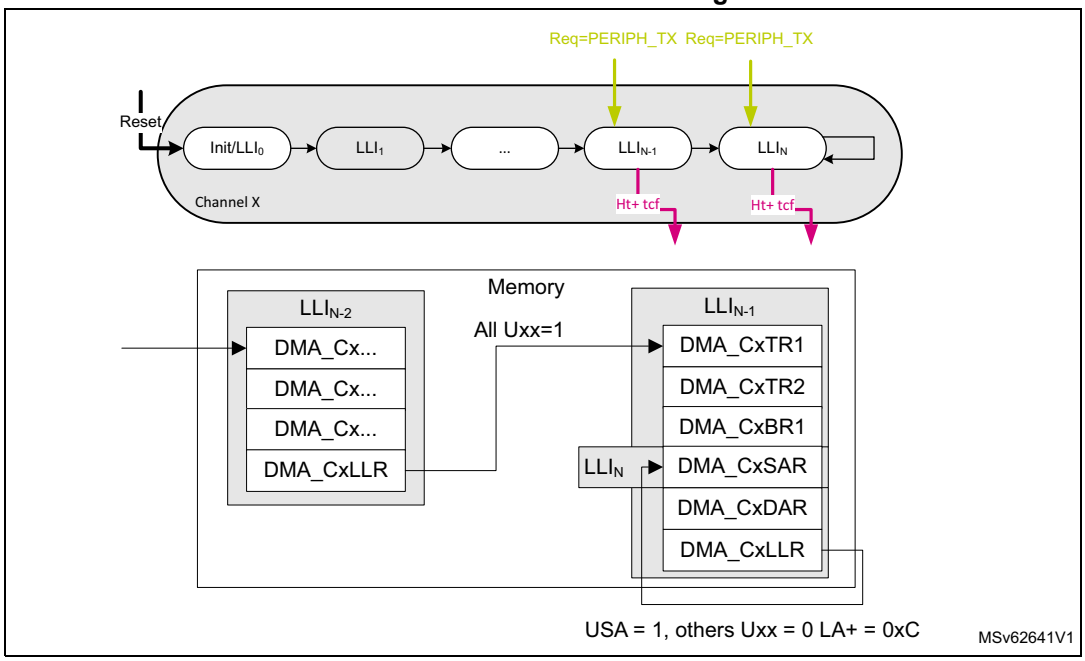


Note: With a 2D addressing channel, a single LLI can be used with HPDMA_CxBR1.BRC[10:0] = 1. The user can program a negative memory block address offset with HDMA_CxBR2 and HDMA_CxBR1, in order to jump back to the memory source or destination start address.

If the circular buffering must be executed after some other transfers over the shared HPDMA channel x, the before-last LLI_{N-1} in the memory is needed to configure the first block transfer. The last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory).

The figure below illustrates this programming with a linear addressing shared HPDMA channel, and a source circular buffer.

Figure 156. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



25.4.15 HPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate an HPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 25.3.6*).

If the related HPDMA channel x is also programmed in peripheral flow-control mode (HPDMA_CxTR2.PFREQ = 1):

- The HPDMA block transfer starts as follows:
  - If HPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the HPDMA hardware.
  - If HPDMA_CxBR1.BNDT[15:0] = 0, the HPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The HPDMA block transfer is completed as soon as the first occurrence of any of the following condition occurs:
  - when HPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on HPDMA_CxTR2. Then the software can read the current number of transferred bytes from the source (HPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (HPDMA_CxSAR[31:0] or HPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with an hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- HPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

25.4.16 HPDMA byte-based padding at the block level

For a given channel x, a LLI_n data transfer consists of a block transfer and is defined by the source block size HPDMA_CxBR1.BNDT[15:0], the size of the block of data to be transfered from the source, in bytes.

During a peripheral-to-memory transfer, a peripheral may be streaming and receiving a given number of bytes (for example, via an I2C, UART/USART/LPUART, ADC, DAC) or half-words, and this block size may not be a multiple of a word or a double-word. It is important to allow the user to program any number of bytes for the HPDMA source block size independently from the source data width and the destination data width, in order to use efficiently the bandwidth of the 32-bit AHB system bus and/or the 64-bit AXI system bus when accessing and writing into memory (embedded SRAM, NVM or external memory). This enables to decrease the bus bandwidth for the memory transfers by a ratio of up to 4 in AHB and up to 8 in AXI. More generally, having the source block size independent from the source data width and the destination data width can be useful also with memory-to-peripheral transfers, and with memory-to-memory transfers (AHB from/to AXI).

A FIFO-based channel is able to perform byte-based padding on the last write(s) up to the fully transferred source block size. Then channel FIFO is empty. Data handling is performed on zero bytes in the same way as on regular data bytes from the source, as explained in *HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment*.

There is byte-based padding at the block level:

1. If source data width is less than the destination data width, and if FIFO queuing and data packing is configured (PAM[1:0] = 1x):

- Padding with zero bytes. For example, if BNDT= 5, 8-bit reads, 32-bit writes:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 000B_4, $B_3B_2B_1B_0$
  - if DBX = 0, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> 0$B_4$00, $B_1B_0B_3B_2$
  - if DBX = 1, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 00$B_4$0, $B_2B_3B_0B_1$
  - if DBX = 1, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> $B_4$000, $B_0B_1B_2B_3$

2. If source data width > destination data width, and if FIFO queuing and data unpacking is configured (PAM[1:0] = 1x):

- Padding with zero bytes.
- On the last read, only the number of LSB(s) up to the block size are written to the FIFO. The last written data to the destination is padded with zero bytes up to the destination data width. For example, if BNDT= 5, 32-bit reads, 16-bit writes:
  - if DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 0B_4, B_3B_2, B_1B_0
  - if DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$, -> $B_4$0, B_2B_3, B_0B_1

3. If source data width = destination data width (and BNDT[15:0] is not a multiple of that value):

- Padding with zero bytes. For example, if BNDT= 6, 32-bit reads, 32-bit writes are configured
  - if SBX = 0, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_5B_4$, $B_3B_2B_1B_0$
  - if SBX = 0, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_5B_400, $B_1B_0B_3B_2$
  - if SBX = 0, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_4B_5$, $B_2B_3B_0B_1$
  - if SBX = 0, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4B_500, $B_0B_1B_2B_3$
  - if SBX = 1, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $000B_4$, $B_3B_1B_2B_0$
  - if SBX = 1, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_400$, $B_2B_0B_3B_1$
  - if SBX = 1, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_40$, $B_1B_3B_0B_2$
  - if SBX = 1, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4000, $B_0B_2B_1B_3$

4. If source data width < destination data width, and if destination data is configured as right-aligned and, either configured with zero padding bytes (PAM[1:0] = 00), or configured with sign extension (PAM[1:0] = 01):

- Padding with zero bytes if PAM[1:0] = 00. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 00:
  - if DBX =0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $000B_4$, $000B_3$, $000B_2$, $000B_1$, $000B_0$
- Padding with sign extension if PAM[1:0] = 01. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 01:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $SSSB_4$, $SSSB_3$, $SSSB_2$, $SSSB_1$, $SSSB_0$

5. If source data width > destination data width, and if destination data is configured as right-aligned (PAM[1:0] = 00), or left-aligned (PAM[1:0] = 01):

- Padding with zero bytes. For example, BNDT = 5, 32-bit reads, 16-bit writes:
  - if SBX = 0, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_1B_0
  - if SBX = 0, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_1
  - if SBX = 0, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_2
  - if SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_2B_3
  - if SBX = 1, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_2B_0
  - If SBX = 1, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_2
  - If SBX = 1, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_1B_3

25.4.17 HPDMA secure/nonsecure channel

The HPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or nonsecure hardware resource, as configured by HPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a bitfield of this channel is forced to return 0, except for HPDMA_SECCFGR, HPDMA_PRIVCFGR, HPDMA_RCFGLOCKR, and HPDMA_CxSEMCR that are readable by a nonsecure agent.
- A nonsecure write access to a bitfield of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the HPDMA data transfer from the source (HPDMA_CxTR1.SSEC), and to the destination (HPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with secure transfers via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a secure bus that reflects HPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each HPDMA channel x.

The HPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure HPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so, and must have HPDMA_CxCR.EN = 0 before the nonsecure software can reprogram HPDMA_CxCR for a next transfer. The secure software can reset not only the channel x (HPDMA_CxCR.RESET = 1), but also the full channel x register file to its reset value.

25.4.18 HPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via HPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a bitfield of this channel is forced to return 0, except for HPDMA_PRIVCFGR, HPDMA_SECCFGR, HPDMA_CxCIDCFGR and HPDMA_CxSEMCR that are readable by an unprivileged agent.
- An unprivileged write access to a bitfield of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB/AXI master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a privileged bus that reflects HPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each HPDMA channel x.

Additionally, the HPDMA generates a privileged illegal access pulse signal on an illegal unprivileged access to a privileged HPDMA register. This signal can be used or not, depending on the product (see the system security section for more details).

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel, or wait until the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer, as an unprivileged software is not allowed to do so, and must have HPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the HPDMA_CxCR for a next transfer. The privileged software can reset not only the channel x (HPDMA_CxCR.RESET = 1), but also the full channel x register file to its reset value.

HPDMA compartmented channel

The HPDMA controller performs bus transfers over its master port, under the control of a resource isolation domain identification named compartment identification (CID), at a channel level. For more details about resource isolation and CID, refer to the RIF section in the product reference manual.

25.4.19 HPDMA error management

The HPDMA can manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as an AHB/AXI single or a burst) to the source or the destination

- The source or destination target reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware). The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1) and the data error (HPDMA_CxSR.DTEF = 1).
- After an HPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After an HPDMA data transfer error, the user must issue a channel reset (set HPDMA_CxCR.RESET) to reset the hardware HPDMA channel data path and the FIFO content, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a HPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware), the channel status register reports an idle state HPDMA_CxSR.IDLEF = 1), and the link error (HPDMA_CxSR.ULEF = 1).
- After an HPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (HPDMA_CxLBAR and HPDMA_CxLLR), including the protection attributes.
- After an HPDMA link error, the user must explicitly write the linked-list register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR,

HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of an HPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (HPDMA_CxCR.EN forced and cleared by the HPDMA hardware), preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1), and a user setting error (HPDMA_CxSR.USEF = 1).
- After an HPDMA user setting error, the user must perform a debug session, taking care of the HPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxLLR.UB1 = 0)
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR1.BNDT[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxSAR[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x =12 to 15, a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxTR3.SAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x =12 to 15, a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR2.BRSAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxDAR[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x =12 to 15, a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxTR3.DAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x =12 to 15, a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR2.BRDAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed double-word source data width and a programmed AHB source allocated port (HPDMA_CxTR1.SDW_LOG2[1:0] = 11 and HPDMA_CxTR1.SAP = 1)
  - a programmed double-word destination data width and a programmed AHB destination allocated port (HPDMA_CxTR1.DDW_LOG2[1:0] = 11 and HPDMA_CxTR1.DAP = 1)
  - a programmed linked-list item LLI_{n+1} with a null data transfer (HPDMA_CxLLR.UB1 = 1 and HPDMA_CxBR1. BNDT = 0)

25.5 HPDMA in debug mode

When the device enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, HPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, HPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

25.6 HPDMA in low-power modes

Table 184. Effect of low-power modes on HPDMA

Mode	Description
Sleep	No effect. HPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of HPDMA registers is kept when entering Stop mode.
Standby	The HPDMA is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 25.3.3* to know which Stop mode is supported.

25.7 HPDMA interrupts

There is one HPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 185. HPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
HPDMA_CHx	Transfer complete	HPDMA_CxCR.TCIE	HPDMA_CxSR.TCF	Writes 1 to HPDMA_CxFCR.TCF
	Half transfer	HPDMA_CxCR.HTIE	HPDMA_CxSR.HTF	Writes 1 to HPDMA_CxFCR.HTF
	Data transfer error	HPDMA_CxCR.DTEIE	HPDMA_CxSR.DTEF	Writes 1 to HPDMA_CxFCR.DTEF
	Update link error	HPDMA_CxCR.ULEIE	HPDMA_CxSR.ULEF	Writes 1 to HPDMA_CxFCR.ULEF
	User setting error	HPDMA_CxCR.USEIE	HPDMA_CxSR.USEF	Writes 1 to HPDMA_CxFCR.USEF
	Suspended	HPDMA_CxCR.SUSPIE	HPDMA_CxSR.SUSPF	Writes 1 to HPDMA_CxFCR.SUSPF
	Trigger overrun	HPDMA_CxCR.TOFIE	HPDMA_CxSR.TOF	Writes 1 to HPDMA_CxFCR.TOF

An HPDMA channel x event can be:

- a transfer complete
- an half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal hpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (HPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (HPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into HPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by HPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer-complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is an half-block transfer, or a half 2D/repeated block transfer, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].

  An half-block transfer occurs when half of the source block size bytes (rounded-up integer of HPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  An half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (HPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details.

Note: The interrupt mode must be used (not the polling mode) to be notified on an half transfer when the write data transaction has been completed over the AXI destination allocated port (written at the destination memory-mapped address), and not just before when has been issued, at HPDMA level, this AXI burst transaction.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of an HPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of an HPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the HPDMA channel programming versus the USEF root causes list (see *Section 25.4.19*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

25.8 HPDMA registers

The HPDMA registers must be accessed with an aligned 32-bit word data access.

25.8.1 HPDMA secure configuration register (HPDMA_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged, or unprivileged, and with any CID.

A write access is ignored at bit level if the corresponding channel x is locked (HPDMA_RCFGLOCKR.LOCKx = 1).

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the SECx bit can be written only by an authorized CID (and secure and privileged) agent. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of an HPDMA channel (when HPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECx**: Secure state of channel x (x = 15 to 0)

0: Nonsecure

1: Secure

25.8.2 HPDMA privileged configuration register (HPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure, and with any CID.

This register can mix secure and nonsecure information. If a channel x is configured as secure (HPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the PRIVx bit can be written only by an authorized CID (and privileged) agent. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

A write access is ignored at bit level if the corresponding channel x is locked (HPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a HPDMA channel (HPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: Privileged state of channel x (x = 15 to 0)
0: Unprivileged
1: Privileged

25.8.3 HPDMA configuration lock register (HPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged and trusted domain CID attributes in order to individually lock, for example at boot time, the secure privileged and CID attributes of any HPDMA channel/resource (to lock the setting of HPDMA_SECCFGR, HPDMA_PRIVCFGR, and HPDMA_CxCIDCFGR for any channel x at, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: Lock of the configuration of HPDMA_SECCFGR.SECx, HPDMA_PRIVCFGR.PRIVx, and HPDMA_CxCIDCFGR until a global HPDMA reset (x = 15 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global HPDMA reset.

0: Secure privilege and CID configuration of the channel x is writable.

1: Secure privilege and CID configuration of the channel x is not writable.

25.8.4 HPDMA nonsecure masked interrupt status register (HPDMA_MISR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with HPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of HPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of HPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in HPDMA_CxFCR.

If a channel x is in secure state (HPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bitfields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: Masked interrupt status of channel x (x = 15 to 0)

0: No interrupt occurred on channel x.

1: An interrupt occurred on channel x.

25.8.5 HPDMA secure masked interrupt status register (HPDMA_SMISR)

Address offset: 0x010

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (HPDMA_SECCFGR.SECx = 1). It is a logical OR of all the HPDMA_CxSR flags, each source flag being enabled by the corresponding HPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding HPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bitfields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: Masked interrupt status of the secure channel x (x = 15 to 0)
0: No interrupt occurred on the secure channel x.
1: An interrupt occurred on the secure channel x.

25.8.6 HPDMA channel x linked-list base address register (HPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x HPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (HPDMA_SECCFGR.SECx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the HPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit HPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **LBA[31:16]**: Linked-list base address of HPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

25.8.7 HPDMA channel x CID register (HPDMA_CxCIDCFGR)

Address offset: 0x054+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This channel-based register can be written by the secure, privileged, and trusted domain CID, in order to individually configure the CID allocation of any HPDMA channel x. If the debug domain feature is activated, a read/write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when the resource configuration of the channel x is unlocked (HPDMA_RCFGLOCKR.LOCKx = 0) and when the channel is disabled (HPDMA_CxCR.EN = 0).

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_WLIST_CID2	SEM_WLIST_CID1	SEM_WLIST_CID0
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[1:0]		Res.	Res.	SEM_EN	CFEN
										rw	rw			rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **SEM_WLIST_CID2**: White-listed CID2 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID2 black-listed in the semaphore-based CID allocation pool
1: CID2 white-listed in the semaphore-based CID allocation pool

Bit 17 **SEM_WLIST_CID1**: White-listed CID1 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID1 black-listed in the semaphore-based CID allocation pool
1: CID1 white-listed in the semaphore-based CID allocation pool

Bit 16 **SEM_WLIST_CID0**: White-listed CID0 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID0 black-listed in the semaphore-based CID allocation pool
1: CID0 white-listed in the semaphore-based CID allocation pool

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **SCID[1:0]**: Allocation of a static/single CID to the channel x (for when the channel x CID configuration is not in semaphore mode)
00: CID0 allocated to the channel x
01: CID1 allocated to the channel x
10: CID2 allocated to the channel x
11: Reserved (write ignored)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEM_EN**: Semaphore mode enable (for the CID allocation policy to the channel x)
0: Semaphore mode disabled. CID allocation policy to the channel x is defined by SCID[1:0].
1: Semaphore mode enabled. CID allocation policy to the channel x is defined by the white-listed allocation pool SEM_WLIST_CIDx and HPDMA_CxSEMCR.SEM_MUTEX.

Note: If SEM_EN = 1 and if a trusted domain or debug domain CID agent clears this bit, then the HPDMA hardware automatically clears the HPDMA_CxSEMCR.SEM_MUTEX.

Bit 0 **CFEN**: CID filtering enable of the channel x
0: CID filtering disabled for when accessing a channel x register/bitfield
1: CID filtering enabled for when accessing a channel x register/bitfield

Note: If CFEN = 1 and if a trusted domain or debug domain CID agent clears this bit, then the HPDMA hardware automatically clears the HPDMA_CxSEMCR.SEM_MUTEX.

25.8.8 HPDMA channel x semaphore control register (HPDMA_CxSEMCR)

Address offset: 0x058+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

When the secure, privileged and trusted domain has set up the channel x in semaphore mode (HPDMA_CxCIDCFGR.SEM_EN = 1), this register is used during run-time by an authorized white-listed CID agent, in order to take and after possibly release the control of the channel x.

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_CCID[1:0]		Res.	Res.	Res.	SEM_MUTEX
										r	r				rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SEM_CCID[1:0]**: Current CID allocated to the channel x (in semaphore mode)

This read-only bitfield is internally updated when a white-listed CID took the control of the channel x, in semaphore mode. If SEM_MUTEX = 0, this same CID also released it.

00: CID0 is the last white-listed CID that took the control of the channel x.

01: CID1 is the last white-listed CID that took the control of the channel x.

10: CID2 is the last white-listed CID that took the control of the channel x.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX**: Mutual exclusion semaphore for the CID allocation of the channel x (in semaphore mode)

If the channel x is in secure state (HPDMA_SECCFGR.SECx = 1), this bit can only be written by a secure agent. If the channel x is in privileged state (HPDMA_PRIVCFGR.PRIVx = 1), this bit can only be written by a privileged agent.

If the channel x is CID-filtered (HPDMA_CxCIDCFGR.CFEN = 1) and in semaphore mode (HPDMA_CIDCFGR.SEM_EN = 1), this bit can only be written either by an authorized (white-listed) CID agent to take the control, or by the same (white-listed) CID agent to release the control.

Condition: write

0: Release the control of the channel x (in semaphore mode) to any white-listed CID.

1: Take the control of the channel x (in semaphore mode), from one of the white-listed CID pool.

Condition: read

0: Channel x CID-free (not currently under the control of any white-listed CID)

1: Channel x CID-allocated (currently taken and under the control of one white-listed CID)

Note: This bit must be written when HPDMA_CxCR.EN = 0. This bit is read-only when HPDMA_CxCR.EN = 1.

When SEM_EN or CFEN bit is cleared in HPDMA_CxSEMCR, the HPDMA hardware automatically clears this SEM_MUTEX bit.

25.8.9 HPDMA channel x flag clear register (HPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag clear
0: No effect
1: Corresponding TOF flag cleared

Bit 13 **SUSPF**: Completed suspension flag clear
0: No effect
1: Corresponding SUSPF flag cleared

Bit 12 **USEF**: User setting error flag clear
0: No effect
1: Corresponding USEF flag cleared

Bit 11 **ULEF**: Update link transfer error flag clear
0: No effect
1: Corresponding ULEF flag cleared

Bit 10 **DTEF**: Data transfer error flag clear
0: No effect
1: Corresponding DTEF flag cleared

Bit 9 **HTF**: Half transfer flag clear
0: No effect
1: Corresponding HTF flag cleared

Bit 8 **TCF**: Transfer complete flag clear
0: No effect
1: Corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

25.8.10 HPDMA channel x status register (HPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAINING	IDLEF
	r	r	r	r	r	r	r							r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **FIFOL[8:0]**: Monitored FIFO level (in bytes)

Number of available bytes in the FIFO.

Note: After having suspended an active transfer, the user may need to read FIFOL[8:0], additionally to HPDMA_CxBR1.BDNT[15:0] and HPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (HPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag

0: No trigger overrun event

1: A trigger overrun event occurred.

Bit 13 **SUSPF**: Completed suspension flag

0: No completed suspension event

1: A completed suspension event occurred.

Bit 12 **USEF**: User setting error flag

0: No user setting error event

1: A user setting error event occurred.

Bit 11 **ULEF**: Update link transfer error flag

0: No update link transfer error event

1: A master bus error event occurred while updating a linked-list register from memory.

Bit 10 **DTEF**: Data transfer error flag

0: No data transfer error event

1: A master bus error event occurred on a data transfer.

Bit 9 **HTF**: Half transfer flag

0: No half transfer event

1: An half transfer event occurred.

An half transfer event is either an half block transfer or an half 2D/repeated block transfer, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).

An half block transfer occurs when half of the bytes of the source block size (rounded up integer of HPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.

An half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (HPDMA_CxBR1.BRC[10:0]+1)/2)) has been transferred to the destination.

Bit 8 **TCF**: Transfer complete flag

0: No transfer complete event

1: A transfer complete event occurred.

A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DRAININGF**: FIFO draining flag

0: No received order to drain the FIFO or channel FIFO draining completed

1: Ongoing channel FIFO draining and channel termination

This flag is asserted after that the order to drain the FIFO is received, and meanwhile that the channel is enabled and draining the FIFO and before that the channel is terminated with a drained FIFO.

This flag is deasserted by hardware when the enabled channel has been ordered to be terminated with FIFO draining, and the hardware effectively terminates.

Bit 0 **IDLEF**: Idle flag

0: Channel not in idle state

1: Channel in idle state

This idle flag is deasserted by hardware when the channel is enabled (HPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).

This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

25.8.11 HPDMA channel x control register (HPDMA_CxCR)

Address offset: 0x064 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAIN FIFO	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw					w	rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** Priority level of the channel x HPDMA transfer versus others
00: Low priority, low weight
01: Low priority, mid weight
10: Low priority, high weight
11: High priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: lLnked-list allocated port
This bit is used to allocate the master port for the update of the HPDMA linked-list registers from the memory.
0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: Channel executed for the full linked-list and completed at the end of the last LLI (HPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0 if present). Then HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 if present.
1: Channel executed once for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if present. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by HPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: Trigger overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **SUSPIE**: cCmpleted suspension interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **USEIE**: User setting error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **ULEIE**: Update link transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **DTEIE**: Data transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **HTIE**: Half transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCIE**: Transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DRAINFIFO:** Empty FIFO
This bit is write only. Writing 0 has no impact.
Writing 1 to RESET in this register causes the hardware to deassert this DRAINFIFO bit, whatever is written into this DRAINFIFO. Else:
Software must write 1 in order to drain the channel FIFO on a suspended channel.
If the software concurrently writes 1 to DRAINFIFO and 1 to SUSP and 0 to reset, the hardware suspends the channel and drains the FIFO by popping out the residual FIFO bytes to the destination (see the programming sequence in *Figure 139*).
0: Write: no action
1: Write: drain FIFO after having suspended the channel

Bit 2 **SUSP:** Suspend
Writing 1 to RESET in this register causes the hardware to deassert this SUSP bit, whatever is written into this SUSP. Else:
Software must write 1 in order to suspend an active channel (a channel with an ongoing HPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 137*.
0: Write: resume channel, read: channel not suspended
1: Write: suspend channel, read: channel suspended

Bit 1 **RESET:** Reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP, DRAINFIFO and EN bits (whatever is written receptively in bit 3 and bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (HPDMA_CxSR.SUSPF = 1 and HPDMA_CxSR.IDLEF = HPDMA_CxCR.EN = 1 and HPDMA_CxSR.DRAININGF = 0)
- channel in disabled state (HPDMA_CxSR.IDLEF = 1 and HPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (HPDMA_CxBR1, HPDMA_CxSAR and HPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 138*).

0: No channel reset

1: Channel reset

Bit 0 **EN:** Enable

Writing 1 to RESET in this register causes the hardware to deassert this EN bit, whatever is written into this bit 0. Else:

This bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: Write: ignored, read: channel disabled

1: Write: enable channel, read: channel enabled

25.8.12 HPDMA channel x transfer register 1 (HPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in HPDMA_PRIVCFGR.PRIVx.

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: Security attribute of the HPDMA transfer to the destination

If HPDMA_SECCFGR.SECx = 1 and the access is secure:

0: HPDMA transfer nonsecure

1: HPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when HPDMA_SECCFGR.SECx = 1. A secure write is ignored when HPDMA_SECCFGR.SECx = 0.

When HPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the HPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: Destination allocated port

This bit is used to allocate the master port for the destination transfer

0: Port 0 (AXI) allocated

1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DWX**: Destination word exchange

If the destination data size is not a double-word, this bit is ignored.

If the destination data size is a double-word and if destination bus is AXI (DAP = 0):

0: No word-based exchanged within double-word

1: The two consecutive (post PAM) words are exchanged in each destination double-word.

Bit 27 **DHX**: Destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word or double-word and if destination bus is AXI (DAP = 0):

0: No half-word-based exchanged within word

1: The two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: Destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0: No byte-based exchange within half-word

1: The two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: Destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Caution: As highlighted in *Section 25.3.2*, the maximum allowed AXI burst length is 16. The user must set DBL_1[5:0] lower or equal to 15 if the destination allocated port is AXI (if DAP = 0).

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol.

If the burst length exceeds 16 on an AHB transfer, or if the burst on an AXI transfer is both with fixed addressing (DINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into bursts of lower length, to be compliant with the AHB or AXI protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: Destination incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The destination address, pointed by HPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: Binary logarithm of the destination data width of a burst, in bytes
00: Byte
01: Half-word (2 bytes)
10: Word (4 bytes)
11: If DAP = 0 (AXI), double-word (8 bytes)
if DAP = 1, user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address HPDMA_CxDAR[2:0] and if present address offset HPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

A burst with a double-word data width must be allocated to the AXI master port, else a user setting error is reported and none transfer is issued.

Bit 15 **SSEC**: Security attribute of the HPDMA transfer from the source
If HPDMA_SECCFGR.SECx = 1 and the access is secure:
0: HPDMA transfer nonsecure
1: HPDMA transfer secure
This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when HPDMA_SECCFGR.SECx = 1. A secure write is ignored when HPDMA_SECCFGR.SECx = 0.
When HPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the HPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: Source allocated port
This bit is used to allocate the master port for the source transfer
0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: Source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word or a double-word, and if source bus is AXI (SAP = 0):
0: No byte-based exchange within the unaligned half-word of each source word
1: The two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: Padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

Condition: PAM_1

00: Source data is transferred as right aligned, padded with 0s up to the destination data width

01: Source data is transferred as right aligned, sign extended up to the destination data width

10-11: Successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: Source data is transferred as right aligned, left-truncated down to the destination data width

01: Source data is transferred as left-aligned, right-truncated down to the destination data width

10-11: Source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: Source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Caution: As highlighted in *Section 25.3.2*, the maximum allowed AXI burst length is 16. The user must set SBL_1[5:0] lower or equal to 15 if the source allocated port is AXI (if SAP = 0).

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or an AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If the burst length exceeds 16 on a AHB transfer, or if the burst on a AXI transfer is both with fixed addressing (SINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: Source incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The source address, pointed by HPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: Binary logarithm of the source data width of a burst in bytes
00: Byte
01: Half-word (2 bytes)
10: Word (4 bytes)
11: If SAP = 0 (AXI), double-word (8 bytes)
if SAP = 1, user setting error reported and no transfer issued
A source burst transfer must have an aligned address with its data width (start address HPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.
A burst with a double-word data width must be allocated to the AXI master port, else a user setting error is reported and none transfer is issued.

25.8.13 HPDMA channel x transfer register 2 (HPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory, if HPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	TRIGSEL[6:0]						
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: Transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: At block level (when HPDMA_CxBR1.BNDT[15:0] = 0): The complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: Channel x = 0 to 11, same as 00; channel x = 12 to 15, at 2D/repeated block level (when HPDMA_CxBR1.BRC[10:0] = 0 and HPDMA_CxBR1.BNDT[15:0] = 0), the complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: At LLI level: The complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer or a 2D/repeated block transfer for channel x = 12 to 15), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: At channel level: The complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address HPDMA_CxLLR.LA[15:2] to zero and clears all the HPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2 if present). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: Trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[6:0].

00: No trigger (masked trigger event)

01: Trigger on the rising edge

10: Trigger on the falling edge

11: Same as 00

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **TRIGSEL[6:0]**: Trigger event input selection

These bits select the trigger event input of the HPDMA transfer (as per *Section 25.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: Trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (HPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, an HPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x = 12 to 15, for each block if a 2D/repeated block is configured with HPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x = 0 to 11, same as 00; channel x = 12 to 15, at 2D/repeated block level, the first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The HPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the HPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1/4-Kbyte boundary address crossing maximum burst length versus AHB/AXI protocol): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 25.3.6 for the list of the implemented channels with this feature.

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in HPDMA control mode. The HPDMA is programmed with HPDMA_CxCBR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The HPDMA block transfer can be early completed by the peripheral itself (see *Section 25.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (HPDMA_CxBR1.BRC[10:0] must be set to 0 if present)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, HPDMA_CxTR1.PAM[1] must be set to 0).

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 25.3.5*).

Bit 10 **DREQ**: Destination hardware request

This bit is ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: Selected hardware request driven by a source peripheral (request signal taken into account by the HPDMA transfer scheduler over the source/read port)

1: Selected hardware request driven by a destination peripheral (request signal taken into account by the HPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (HPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: Software request

This bit is internally taken into account when HPDMA_CxCR.EN is asserted.

0: No software request. The selected hardware request REQSEL[7:0] is taken into account.

1: Software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: Hardware request selection

These bits are ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 25.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active HPDMA channels (HPDMA_CxCR.EN = 1 and HPDMA_CxTR2.SWREQ = 0 for these channels). The HPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

25.8.14 HPDMA channel x block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfield BNDT[15:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If HPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.
- if HPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this bitfield is internally restored to the programmed value.
- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] = 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

25.8.15 HPDMA channel x alternate block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0] after each execution of this final LLI
- If HPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]										
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: At the end of a block transfer, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxBR2.BRDAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a block transfer, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRDAO from the current HPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), HPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the HPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: At the end of a block transfer, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxBR2.BRSAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a block transfer, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRSAO from the current HPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), HPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the HPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This bitfield contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this bitfield becomes read-only. After decrements, this bitfield indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If HPDMA_CxLLR.UB1 = 1, all HPDMA_CxBR1 bitfields are updated by the next LLI in the memory.

– If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.

– if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).

– if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred.

l If DMA_CxTR3.BAM[1:0] = 00, BNDT[15:0] is programmed in number of bytes. The maximum source block size is 64 Kbytes - 1.

l If DMA_CxTR3.BAM[1:0] =01, BNDT[15:0] is programmed as multiple of 16 bytes. The maximum source block size is 1 Mbyte - 16.

l If DMA_CxTR3.BAM[1:0] = 10, BNDT[15:0] is programmed as multiple of 256 bytes. The maximum source block size is 16 Mbytes - 256.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.

- if HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.

- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).

- if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

25.8.16 HPDMA channel x source address register (HPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the source start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.USA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SA[31:0]**: Source address

This bitfield is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (HPDMA_CxTR1.SINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.SBL_1[5:0] and HPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by HPDMA_CxBR1.SDEC and HPDMA_CxTR3.SAO[12:0]

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by HPDMA_CxBR1.BRSDEC and HPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

25.8.17 HPDMA channel x destination address register (HPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the destination start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UDA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DA[31:0]**: destination address

This bitfield is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (HPDMA_CxTR1.DINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.DBL_1[5:0] and HPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by HPDMA_CxBR1.DDEC and HPDMA_CxTR3.DAO[12:0]

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by HPDMA_CxBR1.BRDDEC and HPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

25.8.18 HPDMA channel x transfer register 3 (HPDMA_CxTR3)

Address offset: 0x0A4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRAM[1:0]		Res.	DAO[12:0]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAM[1:0]		Res.	SAO[12:0]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **BRAM[1:0]**: Block repeated alignment mode

00: Repeated block address offset (HPDMA_CxBR2.BRSAO/BRDAO[15:0]) programmed in number of bytes. The absolute maximum repeated block address offset is 64 Kbytes - 1.

01: Repeated block address offset (HPDMA_CxBR2.BRSAO/BRDAO[15:0]) programmed in number of 16 bytes. A repeated block address offset is a multiple of 16 bytes. Absolute maximum repeated block address offset is 1 Mbytes - 16.

10-11: Repeated block address offset (HPDMA_CxBR2.BRSAO/BRDAO[15:0]) programmed in number of 256 bytes. A repeated block address offset is a multiple of 256 bytes. Absolute maximum repeated block address offset is 16 Mbytes - 256.

Bit 29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: Destination address offset increment

The destination address, pointed by HPDMA_CxDAR, is incremented or decremented (depending on HPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:14 **BAM[1:0]**: Block alignment mode

00: Block address offset (HPDMA_CxTR3.SAO/DAO[12:0]) and block size transferred from the source (HPDMA_CxBR1.BNDT[15:0] programmed in number of bytes). Absolute maximum block address offset is 8 Kbytes - 1. Maximum source block size is 64 Kbytes - 1.

01: block address offset (HPDMA_CxTR3.SAO/DAO[12:0]) and block size transferred from the source (HPDMA_CxBR1.BNDT[15:0] programmed as a multiple of 16 bytes). Absolute maximum block address offset is 128 Kbytes - 16. Maximum source block size is 1 Mbytes - 16.

10-11: Block address offset (HPDMA_CxTR3.SAO/DAO[12:0]) and block size transferred from the source (HPDMA_CxBR1.BNDT[15:0] programmed as a multiple of 256 bytes). Absolute maximum block address offset is 2 Mbytes - 256. Maximum source block size is 16 Mbytes - 256.

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: Source address offset increment

The source address, pointed by HPDMA_CxSAR, is incremented or decremented (depending on HPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

25.8.19 HPDMA channel x block register 2 (HPDMA_CxBR2)

Address offset: 0x0A8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UB2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRSAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRDDEC) the current destination address (HPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRSDEC) the current source address (HPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

25.8.20 HPDMA channel x linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory

This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.

0: No HPDMA_CxTR1 update

1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory

This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.

0: No HPDMA_CxTR2 update

1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory

This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0: No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR).

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

25.8.21 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory
This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory
This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory
This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0:No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bit 26 **UT3**: Update HPDMA_CxTR3 from memory
This bit controls the update of HPDMA_CxTR3 from the memory during the link transfer.
0: No HPDMA_CxTR3 update
1: HPDMA_CxTR3 update

Bit 25 **UB2**: Update HPDMA_CxBR2 from memory
This bit controls the update of HPDMA_CxBR2 from the memory during the link transfer.
0: No HPDMA_CxBR2 update
1: HPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL= 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR and HPDMA_CxLLR).
Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

25.8.22 HPDMA hardware configuration 15 register (HPDMA_HWCFGR15)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_CACHE_ATTR15	Res.	Res.	Res.	G_CACHE_ATTR14	Res.	Res.	Res.	G_CACHE_ATTR13	Res.	Res.	Res.	G_CACHE_ATTR12
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_CACHE_ATTR11	Res.	Res.	Res.	G_CACHE_ATTR10	Res.	Res.	Res.	G_CACHE_ATTR9	Res.	Res.	Res.	G_CACHE_ATTR8
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_CACHE_ATTR15:** CACHE attributes support for CHANNEL 15
0: Cache attributes not supported
1: Cache attributes supported

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_CACHE_ATTR14:** CACHE attributes support for CHANNEL 14
0: Cache attributes not supported
1: Cache attributes supported

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_CACHE_ATTR13:** CACHE attributes support for CHANNEL 13
0: Cache attributes not supported
1: Cache attributes supported

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_CACHE_ATTR12:** CACHE attributes support for CHANNEL 12
0: Cache attributes not supported
1: Cache attributes supported

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_CACHE_ATTR11:** CACHE attributes support for CHANNEL 11
0: Cache attributes not supported
1: Cache attributes supported

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_CACHE_ATTR10:** CACHE attributes support for CHANNEL 10
0: Cache attributes not supported
1: Cache attributes supported

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_CACHE_ATTR9:** CACHE attributes support for CHANNEL 9
0: Cache attributes not supported
1: Cache attributes supported

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_CACHE_ATTR8:** CACHE attributes support for CHANNEL 8
0: Cache attributes not supported
1: Cache attributes supported

25.8.23 HPDMA hardware configuration 14 register (HPDMA_HWCFGR14)

Address offset: 0xFBC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_CACHE_ATTR7	Res.	Res.	Res.	G_CACHE_ATTR6	Res.	Res.	Res.	G_CACHE_ATTR5	Res.	Res.	Res.	G_CACHE_ATTR4
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_CACHE_ATTR3	Res.	Res.	Res.	G_CACHE_ATTR2	Res.	Res.	Res.	G_CACHE_ATTR1	Res.	Res.	Res.	G_CACHE_ATTR0
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_CACHE_ATTR7:** CACHE attributes support for CHANNEL 7
0: Cache attributes not supported
1: Cache attributes supported

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_CACHE_ATTR6:** CACHE attributes support for CHANNEL 6
0: Cache attributes not supported
1: Cache attributes supported

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_CACHE_ATTR5:** CACHE attributes support for CHANNEL 5
0: Cache attributes not supported
1: Cache attributes supported

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_CACHE_ATTR4:** CACHE attributes support for CHANNEL 4
0: Cache attributes not supported
1: Cache attributes supported

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_CACHE_ATTR3:** CACHE attributes support for CHANNEL 3
0: Cache attributes not supported
1: Cache attributes supported

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_CACHE_ATTR2:** CACHE attributes support for CHANNEL 2
0: Cache attributes not supported
1: Cache attributes supported

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_CACHE_ATTR1:** CACHE attributes support for CHANNEL 1
0: Cache attributes not supported
1: Cache attributes supported

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_CACHE_ATTR0:** CACHE attributes support for CHANNEL 0
0: Cache attributes not supported
1: Cache attributes supported

25.8.24 HPDMA hardware configuration 13 register (HPDMA_HWCFGR13)

Address offset: 0xFC0

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_PER_CTRL15	Res.	Res.	Res.	G_PER_CTRL14	Res.	Res.	Res.	G_PER_CTRL13	Res.	Res.	Res.	G_PER_CTRL12
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_PER_CTRL11	Res.	Res.	Res.	G_PER_CTRL10	Res.	Res.	Res.	G_PER_CTRL9	Res.	Res.	Res.	G_PER_CTRL8
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_PER_CTRL15:** DMA transfer type for channel 15
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_PER_CTRL14:** DMA transfer type for channel 14
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_PER_CTRL13:** DMA transfer type for channel 13
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_PER_CTRL12:** DMA transfer type for channel 12
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_PER_CTRL11:** DMA transfer type for channel 11
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_PER_CTRL10:** DMA transfer type for channel 10
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_PER_CTRL9:** DMA transfer type for channel 9
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_PER_CTRL8:** DMA transfer type for channel 8
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

25.8.25 HPDMA hardware configuration 12 register (HPDMA_HWCFGR12)

Address offset: 0xFC4

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_PER_CTRL7	Res.	Res.	Res.	G_PER_CTRL6	Res.	Res.	Res.	G_PER_CTRL5	Res.	Res.	Res.	G_PER_CTRL4
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_PER_CTRL3	Res.	Res.	Res.	G_PER_CTRL2	Res.	Res.	Res.	G_PER_CTRL1	Res.	Res.	Res.	G_PER_CTRL0
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_PER_CTRL7:** DMA transfer type for channel 7
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_PER_CTRL6:** DMA transfer type for channel 6
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_PER_CTRL5:** DMA transfer type for channel 5
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_PER_CTRL4:** DMA transfer type for channel 4
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_PER_CTRL3:** DMA transfer type for channel 3
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_PER_CTRL2:** DMA transfer type for channel 2
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_PER_CTRL1:** DMA transfer type for channel 1
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_PER_CTRL0:** DMA transfer type for channel 0
0: No peripheral flow control mode
1: Peripheral flow control mode allowed

25.8.26 HPDMA hardware configuration 11 register (HPDMA_HWCFGR11)

Address offset: 0xFC8

Reset value: 0x0023 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	G_TST_LL_IMPORT	Res.	G_NUM_RESYNC_FFS[2:0]			Res.	G_CID_WIDTH[2:0]		
							r		r	r	r		r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	G_SEC_OPTIONREG[5:0]						Res.	Res.	G_NONSEC_OPTIONREG[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **G_TST_LL_IMPORT:** Master port for the link transfer (DFT purpose only)
0: Port 0
1: Port 1

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **G_NUM_RESYNC_FFS[2:0]:** Number of resynchronization flip-flops, in the range 2 to 6

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **G_CID_WIDTH[2:0]:** CID bus width, in the range 1 to 4

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **G_SEC_OPTIONREG[5:0]:** Secure optional register width, in the range 0 to 32
0: No register
1 to 32: n-bit width secure optional register

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **G_NONSEC_OPTIONREG[5:0]:** nonsecure optional register width, in the range 0 to 32
0: No register
1 to 32: n-bit width nonsecure optional register

25.8.27 HPDMA hardware configuration 10 register (HPDMA_HWCFGR10)

Address offset: 0xFCC

Reset value: 0x2222 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	G_ADDRESSING15[1:0]		Res.	Res.	G_ADDRESSING14[1:0]		Res.	Res.	G_ADDRESSING13[1:0]		Res.	Res.	G_ADDRESSING12[1:0]	
		r	r			r	r			r	r			r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	G_ADDRESSING11[1:0]		Res.	Res.	G_ADDRESSING10[1:0]		Res.	Res.	G_ADDRESSING9[1:0]		Res.	Res.	G_ADDRESSING8[1:0]	
		r	r			r	r			r	r			r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **G_ADDRESSING15[1:0]:** DMA transfer type for channel 15
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **G_ADDRESSING14[1:0]:** DMA transfer type for channel 14
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **G_ADDRESSING13[1:0]:** DMA transfer type for channel 13
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **G_ADDRESSING12[1:0]:** DMA transfer type for channel 12
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **G_ADDRESSING11[1:0]:** DMA transfer type for channel 11
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **G_ADDRESSING10[1:0]:** DMA transfer type for channel 10
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **G_ADDRESSING9[1:0]:** DMA transfer type for channel 9
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **G_ADDRESSING8[1:0]:** DMA transfer type for channel 8
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

25.8.28 HPDMA hardware configuration 9 register (HPDMA_HWCFGR9)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	G_ADDRESSING7[1:0]		Res.	Res.	G_ADDRESSING6[1:0]		Res.	Res.	G_ADDRESSING5[1:0]		Res.	Res.	G_ADDRESSING4[1:0]	
		r	r			r	r			r	r			r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	G_ADDRESSING3[1:0]		Res.	Res.	G_ADDRESSING2[1:0]		Res.	Res.	G_ADDRESSING1[1:0]		Res.	Res.	G_ADDRESSING0[1:0]	
		r	r			r	r			r	r			r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **G_ADDRESSING7[1:0]:** DMA transfer type for channel 7
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **G_ADDRESSING6[1:0]:** DMA transfer type for channel 6
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **G_ADDRESSING5[1:0]:** DMA transfer type for channel 5
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **G_ADDRESSING4[1:0]:** DMA transfer type for channel 4
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **G_ADDRESSING3[1:0]:** DMA transfer type for channel 3
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **G_ADDRESSING2[1:0]:** DMA transfer type for channel 2
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **G_ADDRESSING1[1:0]:** DMA transfer type for channel 1
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **G_ADDRESSING0[1:0]:** DMA transfer type for channel 0
0: Fixed/contiguous block addressing
1: Programmable 2D addressing
2: Programmable 2D addressing with extended offsets and block size

25.8.29 HPDMA hardware configuration 8 register (HPDMA_HWCFGR8)

Address offset: 0xFD4

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_LINKED_LIST15	Res.	Res.	Res.	G_LINKED_LIST14	Res.	Res.	Res.	G_LINKED_LIST13	Res.	Res.	Res.	G_LINKED_LIST12
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_LINKED_LIST11	Res.	Res.	Res.	G_LINKED_LIST10	Res.	Res.	Res.	G_LINKED_LIST9	Res.	Res.	Res.	G_LINKED_LIST8
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_LINKED_LIST15:** DMA transfer type for channel 15
0: No linked-list mode
1: Linked-list mode allowed

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_LINKED_LIST14:** DMA transfer type for channel 14
0: No linked-list mode
1: Linked-list mode allowed

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_LINKED_LIST13:** DMA transfer type for channel 13
0: No linked-list mode
1: Linked-list mode allowed

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_LINKED_LIST12:** DMA transfer type for channel 12
0: No linked-list mode
1: Linked-list mode allowed

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_LINKED_LIST11:** DMA transfer type for channel 11
0: No linked-list mode
1: Linked-list mode allowed

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_LINKED_LIST10:** DMA transfer type for channel 10
0: No linked-list mode
1: Linked-list mode allowed

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_LINKED_LIST9:** DMA transfer type for channel 9
0: No linked-list mode
1: Linked-list mode allowed

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_LINKED_LIST8:** DMA transfer type for channel 8
0: No linked-list mode
1: Linked-list mode allowed

25.8.30 HPDMA hardware configuration 7 register (HPDMA_HWCFGR7)

Address offset: 0xFD8

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_LINKED_LIST7	Res.	Res.	Res.	G_LINKED_LIST6	Res.	Res.	Res.	G_LINKED_LIST5	Res.	Res.	Res.	G_LINKED_LIST4
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_LINKED_LIST3	Res.	Res.	Res.	G_LINKED_LIST2	Res.	Res.	Res.	G_LINKED_LIST1	Res.	Res.	Res.	G_LINKED_LIST0
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_LINKED_LIST7:** DMA transfer type for channel 7
0: No linked-list mode
1: Linked-list mode allowed

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_LINKED_LIST6:** DMA transfer type for channel 6
0: No linked-list mode
1: Linked-list mode allowed

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_LINKED_LIST5:** DMA transfer type for channel 5
0: No linked-list mode
1: Linked-list mode allowed

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_LINKED_LIST4:** DMA transfer type for channel 4
0: No linked-list mode
1: Linked-list mode allowed

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_LINKED_LIST3:** DMA transfer type for channel 3
0: No linked-list mode
1: Linked-list mode allowed

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_LINKED_LIST2:** DMA transfer type for channel 2
0: No linked-list mode
1: Linked-list mode allowed

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_LINKED_LIST1:** DMA transfer type for channel 1
0: No linked-list mode
1: Linked-list mode allowed

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_LINKED_LIST0:** DMA transfer type for channel 0
0: No linked-list mode
1: Linked-list mode allowed

25.8.31 HPDMA hardware configuration 6 register (HPDMA_HWCFGR6)

Address offset: 0xFDC

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_TRANSFERS15	Res.	Res.	Res.	G_TRANSFERS14	Res.	Res.	Res.	G_TRANSFERS13	Res.	Res.	Res.	G_TRANSFERS12
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_TRANSFERS11	Res.	Res.	Res.	G_TRANSFERS10	Res.	Res.	Res.	G_TRANSFERS9	Res.	Res.	Res.	G_TRANSFERS8
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_TRANSFERS15:** DMA transfer type for channel 15
0: Single only
1: Burst and single

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_TRANSFERS14:** DMA transfer type for channel 14
0: Single only
1: Burst and single

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_TRANSFERS13:** DMA transfer type for channel 13
0: Single only
1: Burst and single

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_TRANSFERS12:** DMA transfer type for channel 12
0: Single only
1: Burst and single

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_TRANSFERS11:** DMA transfer type for channel 11
0: Single only
1: Burst and single

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_TRANSFERS10:** DMA transfer type for channel 10
0: Single only
1: Burst and single

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_TRANSFERS9:** DMA transfer type for channel 9
0: Single only
1: Burst and single

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_TRANSFERS8:** DMA transfer type for channel 8
0: Single only
1: Burst and single

25.8.32 HPDMA hardware configuration 5 register (HPDMA_HWCFGR5)

Address offset: 0xFE0

Reset value: 0x1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	G_TRANSFERS7	Res.	Res.	Res.	G_TRANSFERS6	Res.	Res.	Res.	G_TRANSFERS5	Res.	Res.	Res.	G_TRANSFERS4
			r				r				r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_TRANSFERS3	Res.	Res.	Res.	G_TRANSFERS2	Res.	Res.	Res.	G_TRANSFERS1	Res.	Res.	Res.	G_TRANSFERS0
			r				r				r				r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **G_TRANSFERS7:** DMA transfer type for channel 7
0: Single only
1: Burst and single

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **G_TRANSFERS6:** DMA transfer type for channel 6
0: Single only
1: Burst and single

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **G_TRANSFERS5:** DMA transfer type for channel 5
0: Single only
1: Burst and single

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_TRANSFERS4:** DMA transfer type for channel 4
0: Single only
1: Burst and single

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **G_TRANSFERS3:** DMA transfer type for channel 3
0: Single only
1: Burst and single

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **G_TRANSFERS2:** DMA transfer type for channel 2
0: Single only
1: Burst and single

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_TRANSFERS1:** DMA transfer type for channel 1
0: Single only
1: Burst and single

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **G_TRANSFERS0:** DMA transfer type for channel 0
0: Single only
1: Burst and single

25.8.33 HPDMA hardware configuration 4 register (HPDMA_HWCFGR4)

Address offset: 0xFE4

Reset value: 0x7666 5555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	G_FIFO_SIZE15[2:0]			Res.	G_FIFO_SIZE14[2:0]			Res.	G_FIFO_SIZE13[2:0]			Res.	G_FIFO_SIZE12[2:0]		
	r	r	r		r	r	r		r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	G_FIFO_SIZE11[2:0]			Res.	G_FIFO_SIZE10[2:0]			Res.	G_FIFO_SIZE9[2:0]			Res.	G_FIFO_SIZE8[2:0]		
	r	r	r		r	r	r		r	r	r		r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **G_FIFO_SIZE15[2:0]:** FIFO size for the channel 15, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **G_FIFO_SIZE14[2:0]:** FIFO size for the channel 14, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **G_FIFO_SIZE13[2:0]:** FIFO size for the channel 13, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **G_FIFO_SIZE12[2:0]:** FIFO size for the channel 12, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **G_FIFO_SIZE11[2:0]:** FIFO size for the channel 11, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **G_FIFO_SIZE10[2:0]:** FIFO size for the channel 10, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **G_FIFO_SIZE9[2:0]:** FIFO size for the channel 9, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **G_FIFO_SIZE8[2:0]:** FIFO size for the channel 8, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

25.8.34 HPDMA hardware configuration 3 register (HPDMA_HWCFGR3)

Address offset: 0xFE8

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	G_FIFO_SIZE7[2:0]			Res.	G_FIFO_SIZE6[2:0]			Res.	G_FIFO_SIZE5[2:0]			Res.	G_FIFO_SIZE4[2:0]		
	r	r	r		r	r	r		r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	G_FIFO_SIZE3[2:0]			Res.	G_FIFO_SIZE2[2:0]			Res.	G_FIFO_SIZE1[2:0]			Res.	G_FIFO_SIZE0[2:0]		
	r	r	r		r	r	r		r	r	r		r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **G_FIFO_SIZE7[2:0]:** FIFO size for the channel 7, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **G_FIFO_SIZE6[2:0]:** FIFO size for the channel 6, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **G_FIFO_SIZE5[2:0]:** FIFO size for the channel 5, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **G_FIFO_SIZE4[2:0]:** FIFO size for the channel 4, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **G_FIFO_SIZE3[2:0]:** FIFO size for the channel 3, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **G_FIFO_SIZE2[2:0]:** FIFO size for the channel 2, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **G_FIFO_SIZE1[2:0]:** FIFO size for the channel 1, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **G_FIFO_SIZE0[2:0]:** FIFO size for the channel 0, in the range 0 to 7
0: No FIFO. A single source transfer cell is internally registered.
1 to 7: FIFO of 2^{N+1} bytes

25.8.35 HPDMA hardware configuration 2 register (HPDMA_HWCFGR2)

Address offset: 0xFEC

Reset value: 0x0000 7FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	G_MAX_TRIG_ID[6:0]							G_MAX_REQ_ID[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **G_MAX_TRIG_ID[6:0]:**
Maximum trigger event identification (in the range 0 to 127)

Bits 7:0 **G_MAX_REQ_ID[7:0]:**
Maximum peripheral request identification (in the range 0 to 255)

25.8.36 HPDMA hardware configuration 1 register (HPDMA_HWCFGR1)

Address offset: 0xFF0

Reset value: 0x0121 1013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	G_M1_DATA_WIDTH_ENC[1:0]		Res.	Res.	G_M0_DATA_WIDTH_ENC[1:0]		G_MAX_CID[3:0]				Res.	Res.	Res.	G_TRUSTZONE
		r	r			r	r	r	r	r	r				r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	G_NUM_CHANNELS[4:0]					Res.	Res.	Res.	G_PRIVILEGE	Res.	G_MASTER_PORTS[2:0]		
			r	r	r	r	r				r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **G_M1_DATA_WIDTH_ENC[1:0]:** in the range 0 to 2
Master port 1 data bus width:
0: 32-bit
1: 64-bit
2: 128-bit

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **G_M0_DATA_WIDTH_ENC[1:0]:** in the range 0 to 2
Master port 0 data bus width:
0: 32-bit
1: 64-bit
2: 128-bit

Bits 23:20 **G_MAX_CID[3:0]:** in the range 0 to 15
Maximum compartment identification (CID):
0: No CID support, HPDMA not CID-aware
1 to 15: HPDMA is CID-aware. Any channel is a separate CID-aware resource. (support of n+1 CIDs, from CID0 up to CIDn)

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **G_TRUSTZONE:** TrustZone
0: No TrustZone
1: TrustZone support

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **G_NUM_CHANNELS[4:0]:** in the range 2 to 16
Number of HPDMA channels

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **G_PRIVILEGE:** privilege
0: No privilege
1: Privilege support

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **G_MASTER_PORTS[2:0]:** in the range 0 to 5
Master port type:
0: 1x 64-bit AXI (AXI4)
1: 1x AHB: 32-bit port 0 (AHB5)
2: 2x AHB: 32-bit port 0 and 32-bit port 1 (AHB5)
3: 1x AXI 64-bit port 0 (AXI4) + 1x AHB 32-bit port 1 (AHB5)
4: 2x AXI 64-bit port 0 and port 1 (AXI4)
5: 1x AXI 128-bit port 0 (AXI4) + 1x AHB port 1 (AHB5)

25.8.37 HPDMA version register (HPDMA_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0016

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: HPDMA major revision

Bits 3:0 **MINREV[3:0]**: HPDMA minor revision

25.8.38 HPDMA identification register (HPDMA_IPIDR)

Address offset: 0xFF8

Reset value: 0x0010 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: HPDMA identification

25.8.39 HPDMA size identification register (HPDMA_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

This read register identifies the HPDMA as a peripheral with 4 Kbytes of address space.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: size identification

25.8.40 HPDMA register map

Table 186. HPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	HPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	HPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	HPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 - 0x04C	Reserved	Reserved																															
0x050 + 0x80 * x (x = 0 to 15)	HPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x054 + 0x80 * x (x = 0 to 15)	HPDMA_ CxCIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_WLIST_CID2	SEM_WLIST_CID1	SEM_WLIST_CID0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[1:0]		Res.	Res.	SEM_EN	CFEN
	Reset value														0	0	0											0	0			0	0
0x058 + 0x80 * x (x=0 to 15)	HPDMA_ CxSEMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_CCID [1:0]		Res.	Res.	Res.	SEM_MUTEX
	Reset value																											0	0				0
0x05C + 0x080 * x (x=0 to 15)	HPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x060 + 0x080 * x (x=0 to 15)	HPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]									Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAININGF	IDLEF
	Reset value								0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	1
0x064 + 0x080 * x (x=0 to 15)	HPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAINFIFO	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0	0

Table 186. HPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090 + 0x080 * x (x=0 to 15)	HPDMA_CxTR1	DSEC	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x094 + 0x080 * x (x=0 to 15)	HPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	TRIGSEL[6:0]							TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0		0	0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0
0x098 + 0x080 * x (x=0 to 11)	HPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098+ 0x080 * x (x=12 to 15)	HPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C + 0x080 * x (x=0 to 15)	HPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x080 * x (x=0 to 15)	HPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 + 0x080 * x (x=12 to 15)	HPDMA_CxTR3	BRAM[1:0]		Res.	DAO[12:0]													BAM[1:0]		Res.	SAO[12:0]												
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8 + 0x080 * x (x=12 to 15)	HPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC + 0x080 * x (x=0 to 11)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0CC + 0x080 * x (x=12 to 15)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xFB8	HPDMA_ HWCFGR15	Res.	Res.	Res.	G_CACHE_ATTR15	Res.	Res.	Res.	G_CACHE_ATTR14	Res.	Res.	Res.	G_CACHE_ATTR13	Res.	Res.	Res.	G_CACHE_ATTR12	Res.	Res.	Res.	G_CACHE_ATTR11	Res.	Res.	Res.	G_CACHE_ATTR10	Res.	Res.	Res.	G_CACHE_ATTR9	Res.	Res.	Res.	G_CACHE_ATTR8
	Reset value				0				0				0				0				0				0				0				0
0xFBC	HPDMA_ HWCFGR14	Res.	Res.	Res.	G_CACHE_ATTR7	Res.	Res.	Res.	G_CACHE_ATTR6	Res.	Res.	Res.	G_CACHE_ATTR5	Res.	Res.	Res.	G_CACHE_ATTR4	Res.	Res.	Res.	G_CACHE_ATTR3	Res.	Res.	Res.	G_CACHE_ATTR2	Res.	Res.	Res.	G_CACHE_ATTR1	Res.	Res.	Res.	G_CACHE_ATTR0
	Reset value				0				0				0				0				0				0				0				0

Table 186. HPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFC0	HPDMA_ HWCFGR13	Res.	Res.	Res.	G_PER_CTRL15	Res.	Res.	Res.	G_PER_CTRL14	Res.	Res.	Res.	G_PER_CTRL13	Res.	Res.	Res.	G_PER_CTRL12	Res.	Res.	Res.	G_PER_CTRL11	Res.	Res.	Res.	G_PER_CTRL10	Res.	Res.	Res.	G_PER_CTRL9	Res.	Res.	Res.	G_PER_CTRL8
	Reset value				1				1				1				1				1				1				1				1
0xFC4	HPDMA_ HWCFGR12	Res.	Res.	Res.	G_PER_CTRL7	Res.	Res.	Res.	G_PER_CTRL6	Res.	Res.	Res.	G_PER_CTRL5	Res.	Res.	Res.	G_PER_CTRL4	Res.	Res.	Res.	G_PER_CTRL3	Res.	Res.	Res.	G_PER_CTRL2	Res.	Res.	Res.	G_PER_CTRL1	Res.	Res.	Res.	G_PER_CTRL0
	Reset value				1				1				1				1				1				1				1				1
0xFC8	HPDMA_ HWCFGR11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G_TST_LL_IMPOR	Res.	G_NUM_RESYNC _FFS[2:0]			Res.	G_CID_WIDTH [2:0]			Res.	Res.	G_SEC_ OPTIONREG[5:0]						Res.	Res.	G_NONSEC_ OPTIONREG[5:0]					
	Reset value								0		0	1	0		0	1	1			0	0	0	0	0	0			0	0	0	0	0	0
0xFCC	HPDMA_ HWCFGR10	Res.	Res.	G_ADDRESSING15 [1:0]		Res.	Res.	G_ADDRESSING14 [1:0]		Res.	Res.	G_ADDRESSING13 [1:0]		Res.	Res.	G_ADDRESSING12 [1:0]		Res.	Res.	G_ADDRESSING11 [1:0]		Res.	Res.	G_ADDRESSING10 [1:0]		Res.	Res.	G_ADDRESSING 9[1:0]		Res.	Res.	G_ADDRESSING 8[1:0]	
	Reset value			1	0			1	0			1	0			1	0			0	0	0	0	0	0			0	0	0	0	0	0
0xFD0	HPDMA_ HWCFGR9	Res.	Res.	G_ADDRESSING7 [1:0]		Res.	Res.	G_ADDRESSING6 [1:0]		Res.	Res.	G_ADDRESSING5 [1:0]		Res.	Res.	G_ADDRESSING4 [1:0]		Res.	Res.	G_ADDRESSING3 [1:0]		Res.	Res.	G_ADDRESSING2 [1:0]		Res.	Res.	G_ADDRESSING1 [1:0]		Res.	Res.	G_ADDRESSING0 [1:0]	
	Reset value			0	0			0	0			0	0			0	0			0	0			0	0			0	0			0	0
0xFD4	HPDMA_ HWCFGR8	Res.	Res.	Res.	G_LINKED_LIST15	Res.	Res.	Res.	G_LINKED_LIST14	Res.	Res.	Res.	G_LINKED_LIST13	Res.	Res.	Res.	G_LINKED_LIST12	Res.	Res.	Res.	G_LINKED_LIST11	Res.	Res.	Res.	G_LINKED_LIST10	Res.	Res.	Res.	G_LINKED_LIST9	Res.	Res.	Res.	G_LINKED_LIST8
	Reset value				1				1				1				1				1				1				1				1
0xFD8	HPDMA_ HWCFGR7	Res.	Res.	Res.	G_LINKED_LIST7	Res.	Res.	Res.	G_LINKED_LIST6	Res.	Res.	Res.	G_LINKED_LIST5	Res.	Res.	Res.	G_LINKED_LIST4	Res.	Res.	Res.	G_LINKED_LIST3	Res.	Res.	Res.	G_LINKED_LIST2	Res.	Res.	Res.	G_LINKED_LIST1	Res.	Res.	Res.	G_LINKED_LIST0
	Reset value				1				1				1				1				1				1				1				1
0xFDC	HPDMA_ HWCFGR6	Res.	Res.	Res.	G_TRANSFERS15	Res.	Res.	Res.	G_TRANSFERS14	Res.	Res.	Res.	G_TRANSFERS13	Res.	Res.	Res.	G_TRANSFERS12	Res.	Res.	Res.	G_TRANSFERS11	Res.	Res.	Res.	G_TRANSFERS10	Res.	Res.	Res.	G_TRANSFERS9	Res.	Res.	Res.	G_TRANSFERS8
	Reset value				1				1				1				1				1				1				1				1

Table 186. HPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	HPDMA_ HWCFGR5	Res.	Res.	Res.	G_TRANSFERS7	Res.	Res.	Res.	G_TRANSFERS6	Res.	Res.	Res.	G_TRANSFERS5	Res.	Res.	Res.	G_TRANSFERS4	Res.	Res.	Res.	G_TRANSFERS3	Res.	Res.	Res.	G_TRANSFERS2	Res.	Res.	Res.	G_TRANSFERS1	Res.	Res.	Res.	G_TRANSFERS0
	Reset value				1				1				1				1				1				1				1				1
0xFE4	HPDMA_ HWCFGR4		G_FIFO_SIZE15 [2:0]				G_FIFO_SIZE14 [2:0]				G_FIFO_SIZE13 [2:0]				G_FIFO_SIZE12 [2:0]				G_FIFO_SIZE11 [2:0]				G_FIFO_SIZE10 [2:0]				G_FIFO_SIZE9 [2:0]				G_FIFO_SIZE8 [2:0]		
	Reset value		1	1	1		1	1	0		1	1	0		1	1	0		1	0	1		1	0	1		1	0	1		1	0	1
0xFE8	HPDMA_ HWCFGR3		G_FIFO_SIZE7 [2:0]				G_FIFO_SIZE6 [2:0]				G_FIFO_SIZE5 [2:0]				G_FIFO_SIZE4 [2:0]				G_FIFO_SIZE3 [2:0]				G_FIFO_SIZE2 [2:0]				G_FIFO_SIZE1 [2:0]				G_FIFO_SIZE0 [2:0]		
	Reset value		1	0	0		1	0	0		1	0	0		1	0	0		1	0	0		1	0	0		1	0	0		1	0	0
0xFEC	HPDMA_ HWCFGR2																		G_MAX_TRIG_ID [6:0]							G_MAX_REQ_ID [7:0]							
	Reset value																		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xFF0	HPDMA_ HWCFGR1			G_M1_DATA_ WIDTH_ENC[1:0]				G_M0_DATA_ WIDTH_ENC[1:0]			G_MAX_CID[3:0]						G_TRUSTZONE				G_NUM_CHANNELS [4:0]							G_PRIVILEGE			G_MASTER_PORTS [2:0]		
	Reset value			0	0			0	1	0	0	1	0				1				1	0	0	0	0				1		0	1	1
0xFF4	HPDMA_VERR																									MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFF8	HPDMA_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0xFFC	HPDMA_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

26 Interrupt list

26.1 NVIC interrupts

Table 187. STM32MP21xx interrupt mapping for Cortex-M33

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
-	-	-	-	Reserved.	0000 0000	-	-
-	-4	fixed	Reset	Reset.	0000 0004	-	-
-	-2	fixed	NMI	Reserved.	0000 0008	-	-
-	-3 or -1	fixed	Secure HardFault	Secure HardFault.	0000 000C	-	-
-	-1	fixed	M33 nonsecure HardFault	Nonsecure Hard fault. All classes of fault.	0000 000C	-	-
-	0	settable	M33 MemManage	Memory management.	0000 0010	-	-
-	1	settable	M33 BusFault	Pre-fetch fault, memory access fault.	0000 0014	-	-
-	2	settable	UsageFault	Undefined instruction or illegal state.	0000 0018	-	-
-	3	settable	SecureFault	SecureFault0.	0000 001C	-	-
-	-	-	-	Reserved.	0000 0020 - 0000 0028	-	-
-	4	settable	SVCall	System service call via SWI instruction.	0000 002C	-	-
-	5	settable	M33 Debug Monitor	Debug Monitor.	0000 0030	-	-
-	-	-	-	Reserved.	0000 0034	-	-
-	6	settable	PendSV	Pendable request for system service.	0000 0038	-	-
-	7	settable	Systick	System tick timer.	0000 003C	-	-
0	8	settable	PVD	PVD detector through EXTI.	00000040	16	-
1	9	settable	PVM	PVM detector through EXTI.	00000044	17	-
2	10	settable	IWDG3	Independent Watchdog 3 early wake interrupt.	00000048	-	(41)
3	11	settable	IWDG4	Independent watchdog 4 early wake interrupt.	0000004C	-	(42)
4	12	settable	IWDG1_RST	Independent watchdog 1 reset through EXTI.	00000050	-	34
5	13	settable	IWDG2_RST	Independent watchdog 2 reset through EXTI.	00000054	-	35

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
6	14	settable	IWDG4_RST	Independent watchdog 4 reset through EXTI.	00000058	-	37
7	15	settable	WWDG1	Window watchdog 1 early wakeup interrupt.	0000005C	-	-
8	16	settable	-	-	00000060	-	-
9	17	settable	-	-	00000064	-	-
10	18	settable	TAMP	Tamper interrupt (include LSECSS interrupts).	00000068	-	(18)
11	19	settable	RTC	RTC global interrupt.	0000006C	-	(19)
12	20	settable	TAMP_S	Tamper secure interrupt (include LSECSS interrupts).	00000070	-	(23)
13	21	settable	RTC_S	RTC global secure interrupt.	00000074	-	(24)
14	22	settable	RCC	RCC global interrupt.	00000078	-	-
15	23	settable	EXT2_0	EXT2 line 0 interrupt.	0000007C	-	0
16	24	settable	EXT2_1	EXT2 line 1 interrupt.	00000080	-	1
17	25	settable	EXT2_2	EXT2 line 2 interrupt.	00000084	-	2
18	26	settable	EXT2_3	EXT2 line 3 interrupt.	00000088	-	3
19	27	settable	EXT2_4	EXT2 line 4 interrupt.	0000008C	-	4
20	28	settable	EXT2_5	EXT2 line 5 interrupt.	00000090	-	5
21	29	settable	EXT2_6	EXT2 line 6 interrupt.	00000094	-	6
22	30	settable	EXT2_7	EXT2 line 7 interrupt.	00000098	-	7
23	31	settable	EXT2_8	EXT2 line 8 interrupt.	0000009C	-	8
24	32	settable	EXT2_9	EXT2 line 9 interrupt.	000000A0	-	9
25	33	settable	EXT2_10	EXT2 line 10 interrupt.	000000A4	-	10
26	34	settable	EXT2_11	EXT2 line 11 interrupt.	000000A8	-	11
27	35	settable	EXT2_12	EXT2 line 12 interrupt.	000000AC	-	12
28	36	settable	EXT2_13	EXT2 line 13 interrupt.	000000B0	-	13
29	37	settable	EXT2_14	EXT2 line 14 interrupt.	000000B4	-	14
30	38	settable	EXT2_15	EXT2 line 15 interrupt.	000000B8	-	15
31	39	settable	HPDMA1_Channel0	HPDMA1 channel0 interrupt.	000000BC	-	-
32	40	settable	HPDMA1_Channel1	HPDMA1 channel1 interrupt.	000000C0	-	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
33	41	settable	HPDMA1_Channel2	HPDMA1 channel2 interrupt.	000000C4	-	-
34	42	settable	HPDMA1_Channel3	HPDMA1 channel3 interrupt.	000000C8	-	-
35	43	settable	HPDMA1_Channel4	HPDMA1 channel4 interrupt.	000000CC	-	-
36	44	settable	HPDMA1_Channel5	HPDMA1 channel5 interrupt.	000000D0	-	-
37	45	settable	HPDMA1_Channel6	HPDMA1 channel6 interrupt.	000000D4	-	-
38	46	settable	HPDMA1_Channel7	HPDMA1 channel7 interrupt.	000000D8	-	-
39	47	settable	HPDMA1_Channel8	HPDMA1 channel8 interrupt.	000000DC	-	-
40	48	settable	HPDMA1_Channel9	HPDMA1 channel9 interrupt.	000000E0	-	-
41	49	settable	HPDMA1_Channel10	HPDMA1 channel10 interrupt.	000000E4	-	-
42	50	settable	HPDMA1_Channel11	HPDMA1 channel11 interrupt.	000000E8	-	-
43	51	settable	HPDMA1_Channel12	HPDMA1 channel12 interrupt.	000000EC	-	-
44	52	settable	HPDMA1_Channel13	HPDMA1 channel13 interrupt.	000000F0	-	-
45	53	settable	HPDMA1_Channel14	HPDMA1 channel14 interrupt.	000000F4	-	-
46	54	settable	HPDMA1_Channel15	HPDMA1 channel15 interrupt.	000000F8	-	-
47	55	settable	HPDMA2_Channel0	HPDMA2 channel0 interrupt.	000000FC	-	-
48	56	settable	HPDMA2_Channel1	HPDMA2 channel1 interrupt.	00000100	-	-
49	57	settable	HPDMA2_Channel2	HPDMA2 channel2 interrupt.	00000104	-	-
50	58	settable	HPDMA2_Channel3	HPDMA2 channel3 interrupt.	00000108	-	-
51	59	settable	HPDMA2_Channel4	HPDMA2 channel4 interrupt.	0000010C	-	-
52	60	settable	HPDMA2_Channel5	HPDMA2 channel5 interrupt.	00000110	-	-
53	61	settable	HPDMA2_Channel6	HPDMA2 channel6 interrupt.	00000114	-	-
54	62	settable	HPDMA2_Channel7	HPDMA2 channel7 interrupt.	00000118	-	-
55	63	settable	HPDMA2_Channel8	HPDMA2 channel8 interrupt.	0000011C	-	-
56	64	settable	HPDMA2_Channel9	HPDMA2 channel9 interrupt.	00000120	-	-
57	65	settable	HPDMA2_Channel10	HPDMA2 channel10 interrupt.	00000124	-	-
58	66	settable	HPDMA2_Channel11	HPDMA2 channel11 interrupt.	00000128	-	-
59	67	settable	HPDMA2_Channel12	HPDMA2 channel12 interrupt.	0000012C	-	-
60	68	settable	HPDMA2_Channel13	HPDMA2 channel13 interrupt.	00000130	-	-
61	69	settable	HPDMA2_Channel14	HPDMA2 channel14 interrupt.	00000134	-	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
62	70	settable	HPDMA2_Channel15	HPDMA2 channel15 interrupt.	00000138	-	-
63	71	settable	HPDMA3_Channel0	HPDMA3 channel0 interrupt.	0000013C	-	-
64	72	settable	HPDMA3_Channel1	HPDMA3 channel1 interrupt.	00000140	-	-
65	73	settable	HPDMA3_Channel2	HPDMA3 channel2 interrupt.	00000144	-	-
66	74	settable	HPDMA3_Channel3	HPDMA3 channel3 interrupt.	00000148	-	-
67	75	settable	HPDMA3_Channel4	HPDMA3 channel4 interrupt.	0000014C	-	-
68	76	settable	HPDMA3_Channel5	HPDMA3 channel5 interrupt.	00000150	-	-
69	77	settable	HPDMA3_Channel6	HPDMA3 channel6 interrupt.	00000154	-	-
70	78	settable	HPDMA3_Channel7	HPDMA3 channel7 interrupt.	00000158	-	-
71	79	settable	HPDMA3_Channel8	HPDMA3 channel8 interrupt.	0000015C	-	-
72	80	settable	HPDMA3_Channel9	HPDMA3 channel9 interrupt.	00000160	-	-
73	81	settable	HPDMA3_Channel10	HPDMA3 channel10 interrupt.	00000164	-	-
74	82	settable	HPDMA3_Channel11	HPDMA3 channel11 interrupt.	00000168	-	-
75	83	settable	HPDMA3_Channel12	HPDMA3 channel12 interrupt.	0000016C	-	-
76	84	settable	HPDMA3_Channel13	HPDMA3 channel13 interrupt.	00000170	-	-
77	85	settable	HPDMA3_Channel14	HPDMA3 channel14 interrupt.	00000174	-	-
78	86	settable	HPDMA3_Channel15	HPDMA3 channel15 interrupt.	00000178	-	-
79	87	settable	ICACHE	ICACHE interrupt.	0000017C	-	-
80	88	settable	DCACHE	DCACHE interrupt.	00000180	-	-
81	89	settable	ADC1	ADC1 interrupt.	00000184	-	-
82	90	settable	ADC2	ADC2 interrupt.	00000188	-	-
83	91	settable	FDCAN_CAL	FDCAN CCU interrupt.	0000018C	-	-
84	92	settable	FDCAN1_IT0	FDCAN1 interrupt 0.	00000190	-	-
85	93	settable	FDCAN2_IT0	FDCAN2 interrupt 0.	00000194	-	-
86	94	settable	FDCAN1_IT1	FDCAN1 interrupt 1.	00000198	-	-
87	95	settable	FDCAN2_IT1	FDCAN2 interrupt 1.	0000019C	-	-
88	96	settable	TIM1_BRK	TIM1 break interrupt.	000001A0	-	-
89	97	settable	TIM1_UP	TIM1 update interrupt.	000001A4	-	-
90	98	settable	TIM1_TRG_COM	TIM1 trigger and commutation interrupts.	000001A8	-	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
91	99	settable	TIM1_CC	TIM1 capture compare interrupt.	000001AC	-	-
92	100	settable	TIM2	TIM2 global interrupt.	000001B0	-	-
93	101	settable	TIM3	TIM3 global interrupt.	000001B4	-	-
94	102	settable	TIM4	TIM4 global interrupt.	000001B8	-	-
95	103	settable	I2C1	I2C1 global interrupt.	000001BC	(21)	-
96	104	settable	I3C1	I3C1 global interrupt.	000001C0	(74)	-
97	105	settable	I2C2	I2C2 global interrupt.	000001C4	(22)	-
98	106	settable	I3C2	I3C2 global interrupt.	000001C8	(75)	-
99	107	settable	SPI1	SPI1 global interrupt.	000001CC	(36)	-
100	108	settable	SPI2	SPI2 global interrupt.	000001D0	(37)	-
101	109	settable	USART1	USART1 global interrupt.	000001D4	(26)	-
102	110	settable	USART2	USART2 global interrupt.	000001D8	(27)	-
103	111	settable	USART3	USART3 global interrupt.	000001DC	(28)	-
104	112	settable	-	Reserved.	000001E0	-	-
105	113	settable	TIM8_BRK	TIM8 break interrupt.	000001E4	-	-
106	114	settable	TIM8_UP	TIM8 update interrupt.	000001E8	-	-
107	115	settable	TIM8_TRG_COM	TIM8 trigger & commutation interrupt.	000001EC	-	-
108	116	settable	TIM8_CC	TIM8 capture compare interrupt.	000001F0	-	-
109	117	settable	FMC	FMC global interrupt.	000001F4	-	-
110	118	settable	SDMMC1	SDMMC1 global interrupt.	000001F8	-	-
111	119	settable	TIM5	TIM5 global interrupt.	000001FC	-	-
112	120	settable	SPI3	SPI3 global interrupt.	00000200	(38)	-
113	121	settable	UART4	UART4 global interrupt.	00000204	(30)	-
114	122	settable	UART5	UART5 global interrupt.	00000208	(31)	-
115	123	settable	TIM6	TIM6 global interrupt.	0000020C	-	-
116	124	settable	TIM7	TIM7 global interrupt.	00000210	-	-
117	125	settable	ETH1_SBD	ETH1 global interrupt.	00000214	-	-
118	126	settable	ETH1_PMT	ETH1 wakeup interrupt (PMT).	00000218	(68)	-
119	127	settable	ETH1_LPI	ETH1 LPI interrupt.	0000021C	-	

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
120	128	settable	ETH2_SBD	ETH2 global interrupt.	00000220	-	-
121	129	settable	ETH2_PMT	ETH2 wakeup interrupt (PMT).	00000224	(70)	-
122	130	settable	ETH2_LPI	ETH2 LPI interrupt.	00000228	-	-
123	131	settable	USART6	USART6 global interrupt.	0000022C	(29)	-
124	132	settable	I2C3	I2C3 global interrupt.	00000230	(23)	-
125	133	settable	I3C3	I3C3 global interrupt.	00000234	(76)	-
126	134	settable	USBH_EHCI	USB host EHCI interrupt.	00000238		-
127	135	settable	USBH_OHCI	USB host OHCI interrupt.	0000023C		-
128	136	settable	DCMI_PSSI	DCMI & PSSI global interrupt.	00000240	-	-
129	137	settable	CSI2HOST	CSI2 host controller interrupt .	00000244	-	-
130	138	settable	-	Reserved.	00000248	-	-
131	139	settable	CRYP1	Crypto1 interrupt.	0000024C	-	-
132	140	settable	HASH1	Hash1 interrupt.	00000250	-	-
133	141	settable	PKA	PKA interrupt.	00000254	-	-
134	142	Settable	FPU	FPU global interrupt.	00000258	-	-
135	143	settable	UART7	UART7 global interrupt.	0000025C	(32)	-
136	144	settable	LPUART1	LPUART1 global interrupt.	00000260	-	(26)
137	145	settable	SPI4	SPI4 global interrupt.	00000264	(39)	-
138	146	settable	SPI5	SPI5 global interrupt.	00000268	(40)	-
139	147	settable	SPI6	SPI6 global interrupt.	0000026C	(41)	-
140	148	settable	SAI1	SAI1 global interrupt.	00000270	-	-
141	149	settable	LTDC_IRQ	LTDC global interrupt.	00000274	-	-
142	150	settable	LTDC_ER_IRQ	LTDC global error interrupt.	00000278	-	-
143	151	settable	LTDC_SEC_IRQ	LTDC security global interrupt.	0000027C	-	-
144	152	settable	LTDC_SEC_ER_IRQ	LTDC security global error interrupt.	00000280	-	-
145	153	settable	SAI2	SAI2 global interrupt.	00000284	-	-
146	154	settable	OCTOSPI1	OCTOSPI1 global interrupt.	00000288	-	-
147	155	settable	-	Reserved.	0000028C	-	-
148	156	settable	OTFDEC1	OTFDEC1 interrupt.	00000290	-	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
149	157	settable	LPTIM1	LPTIMER1 global interrupt.	00000294	(47)	-
150	158	settable	USBH WAKEUP	USBH remote wakeup from USB2PHY1.	00000298	43	-
151	159	settable	SPDIFRX	SPDIFRX global interrupt.	0000029C	-	-
152	160	settable	IPCC1_RX	Mailbox 1 RX occupied interrupt.	000002A0	(60)	-
153	161	settable	IPCC1_TX	Mailbox 1 TX free interrupt.	000002A4	(60)	-
154	162	settable	IPCC1_RX_S	Mailbox 1 RX occupied secure interrupt.	000002A8	(62)	-
155	163	settable	IPCC1_TX_S	Mailbox 1 TX free secure interrupt.	000002AC	(62)	-
156	164	settable	SAES	Secure AES.	000002B0	-	-
157	165	settable	CRYP2	Crypto2 interrupt.	000002B4	-	-
158	166	settable	OTG WAKEUP	USB2 DR remote wakeup from USB2PHY2.	000002B8	44	-
159	167	settable	-	Reserved.	000002BC	-	-
160	168	settable	MDF1_FLT0	MDF1 filter0 interrupt.	000002C0	-	-
161	169	settable	MDF1_FLT1	MDF1 filter1 interrupt.	000002C4	-	-
162	170	settable	MDF1_FLT2	MDF1 filter2 interrupt.	000002C8	-	-
163	171	settable	MDF1_FLT3	MDF1 filter3 interrupt.	000002CC	-	-
164	172	settable	SAI3	SAI3 global interrupt.	000002D0	-	-
165	173	settable	TIM15	TIM15 global interrupt.	000002D4	-	-
166	174	settable	TIM16	TIM16 global interrupt.	000002D8	-	-
167	175	settable	TIM17	TIM17 global interrupt.	000002DC	-	-
168	176	settable	TIM12	TIM12 global interrupt.	000002E0	-	-
169	177	settable	SDMMC2	SDMMC2 global interrupt.	000002E4	-	-
170	178	settable	DCMIPP	DCMIPP global interrupt.	000002E8	-	-
171	179	settable	nCTIIRQ1	Cortex-M33 CTI interrupt 1.	000002EC	-	-
172	180	settable	nCTIIRQ2	Cortex-M33 CTI interrupt 2.	000002F0	-	-
173	181	settable	TIM13	TIM13 global interrupt.	000002F4	-	-
174	182	settable	TIM14	TIM14 global interrupt.	000002F8	-	-
175	183	settable	TIM10	TIM10 global interrupt.	000002FC	-	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
176	184	settable	RNG1	RNG1 interrupt.	00000300	-	-
177	185	settable	-	-	00000304	-	-
178	186	settable	HASH2	HASH2 interrupt.	00000308		-
179	187	settable	SDMMC3	SDMMC3 global interrupt.	0000030C	-	-
180	188	settable	LPTIM2	LPTIMER2 global interrupt.	00000310	(48)	-
181	189	settable	LPTIM3	LPTIMER3 global interrupt.	00000314	-	(29)
182	190	settable	LPTIM4	LPTIMER4 global interrupt.	00000318	-	(30)
183	191	settable	LPTIM5	LPTIMER5 global interrupt.	0000031C	-	(31)
184	192	settable	-	Reserved.	00000320	-	-
185	193	settable	CPU1_SEV	Cortex-A35 send event through EXTI.	00000324	65	-
186	194	settable	RCC_WAKEUP	RCC CPU2 wakeup interrupt.	00000328	-	-
187	195	settable	SAI4	SAI4 global interrupt.	0000032C	-	-
188	196	settable	DTS	Temperature sensor interrupt.	00000330	(72)	-
189	197	settable	TIM11	TIM11 global interrupt.	00000334	-	-
190	198	settable	CPU2_WAKEUP_PIN	Interrupt for all 6 wakeup enabled by CPU2.	00000338	(52, 53, 54, 55, 56, 57)	-
191	199	settable	OTG	USB2 DR global interrupt.	0000033C	-	-
192	200	settable	RNG2	RNG2 interrupt.	00000340		-
193	201	settable	-	Reserved.	00000344	(46)	-
194	202	settable	Reserved	Reserved.	00000348		-
195	203	settable	SERF	SERF global interrupt.	0000034C	-	-
196	204	settable	-	Reserved.	00000350	-	-
197	205	settable	RAMCFG	RAM configuration global interrupt.	00000354	-	-
198	206	settable	DDRCTRL	DDRCTRL interrupt.	00000358	-	-
199	207	settable	DDRPHYC	DDRPHY interrupt (active low).	0000035C	-	-
200	208	settable	DFI_ERR	DDRPHY DFI error interrupt.	00000360	-	-
201	209	settable	IAC	RIF illegal access controller interrupt.	00000364	-	(50)

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
202	210	settable	VDDCPU_VD	VDDCPU voltage detector.	00000368	-	51
203	211	settable	VDDCORE_VD	VDDCORE voltage detector.	0000036C	-	52
204	212	settable	-	-	00000370	-	-
205	213	settable	HPDMA1 WKUP	HPDMA1 channel 0 to 15 wakeup.	00000374	(80)	-
206	214	settable	HPDMA2 WKUP	HPDMA2 channel 0 to 15 wakeup.	00000378	(81)	-
207	215	settable	HPDMA3 WKUP	HPDMA3 channel 0 to 15 wakeup.	0000037C	(82)	-
208	216	settable	DBG_AUTH_HOST	debugger has written DBG_AUTH_HOST register.	00000380	-	-
209	217	settable	DBG_AUTH_ACK	debugger has read DBG_AUTH_ACK register.	00000384	-	-
210	218	settable	RCC_HSI_FMON	HSI frequency monitor.	00000388	(19)	-
211	219	settable	RCC_MSI_FMON	MSI frequency monitor.	0000038C	(18)	-
212 to 267	221 to 275	settable	-	-	000004B0 - 0000050C	-	-
268	276	settable	EXTI1_0	EXTI1 Line 0 interrupt	00000470	0	-
269	277	settable	EXTI1_1	EXTI1 Line 1 interrupt	00000474	1	-
270	278	settable	EXTI1_2	EXTI1 Line 2 interrupt	00000478	2	-
271	279	settable	EXTI1_3	EXTI1 Line 3 interrupt	0000047C	3	-
272	280	settable	EXTI1_4	EXTI1 Line 4 interrupt	00000480	4	-
273	281	settable	EXTI1_5	EXTI1 Line 5 interrupt	00000484	5	-
274	282	settable	EXTI1_6	EXTI1 Line 6 interrupt	00000488	6	-
275	283	settable	EXTI1_7	EXTI1 Line 7 interrupt	0000048C	7	-
276	284	settable	EXTI1_8	EXTI1 Line 8 interrupt	00000490	8	-
277	285	settable	EXTI1_9	EXTI1 Line 9 interrupt	00000494	9	-
278	286	settable	EXTI1_10	EXTI1 Line 10 interrupt	00000498	10	-
279	287	settable	EXTI1_11	EXTI1 Line 11 interrupt	0000049C	11	-
280	288	settable	EXTI1_12	EXTI1 Line 12 interrupt	000004A0	12	-
281	289	settable	EXTI1_13	EXTI1 Line 13 interrupt	000004A4	13	-
282	290	settable	EXTI1_14	EXTI1 Line 14 interrupt	000004A8	14	-
283	291	settable	EXTI1_15	EXTI1 Line 15 interrupt	000004AC	15	-

Table 187. STM32MP21xx interrupt mapping for Cortex-M33 (continued)

Position	Priority	Type	Acronym	Description	Address (hexadec. values)	EXTI1 event	EXTI2 event
284 to 307	292 to 315	settable	-	-	000004B0 to 0000050C	-	-
308	316	settable	-	Reserved.	00000510	-	-
309	317	settable	-	-	00000514	-	-
310	318	settable	DDRPERFM	DDR performance monitor interrupt.	00000518	-	-
311 to 319	323 to 327	settable	-	-	00000530 - 0000053C	-	-

26.2 GIC interrupts

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
0	0	SGI0	Software generated interrupt 0 (recommended nonsecure).	-	-
1	1	SGI1	Software generated interrupt 1 (recommended nonsecure).	-	-
2	2	SGI2	Software generated interrupt 2 (recommended nonsecure).	-	-
3	3	SGI3	Software generated interrupt 3 (recommended nonsecure).	-	-
4	4	SGI4	Software generated interrupt 4 (recommended nonsecure).	-	-
5	5	SGI5	Software generated interrupt 5 (recommended nonsecure).	-	-
6	6	SGI6	Software generated interrupt 6 (recommended nonsecure).	-	-
7	7	SGI7	Software generated interrupt 7 (recommended nonsecure).	-	-
8	8	SGI8	Software generated interrupt 8 (recommended secure).	-	-
9	9	SGI9	Software generated interrupt 9 (recommended secure).	-	-
10	10	SGI10	Software generated interrupt 10 (recommended secure).	-	-
11	11	SGI11	Software generated interrupt 11 (recommended secure).	-	-
12	12	SGI12	Software generated interrupt 12 (recommended secure).	-	-
13	13	SGI13	Software generated interrupt 13 (recommended secure).	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
14	14	SGI14	Software generated interrupt 14 (recommended secure).	-	-
15	15	SGI15	Software generated interrupt 15 (recommended secure).	-	-
-	16 to 24	-	Not implemented.	-	-
-	25	-	Reserved.	-	-
-	26	PPI26	Hypervisor timer event.	-	-
-	27	PPI27	Virtual timer event.	-	-
-	28	-	Reserved.	-	-
-	29	PPI29	Secure physical timer event.	-	-
-	30	PPI30	Nonsecure physical timer event.	-	-
-	31	-	Reserved.	-	-
SPI (active high level sensitive or rising edge triggered)				EXTI1 Event	EXTI2 Event
0	32	PVD	PVD detector through EXTI.	16	-
1	33	PVM	PVM detector through EXTI.	17	-
2	34	IWDG1	Independent watchdog 1 early wake interrupt.	-	(39)
3	35	IWDG2	Independent watchdog 2 early wake interrupt.	-	(40)
4	36	IWDG3_RST	Independent watchdog 3 reset through EXTI.	-	36
5	37	IWDG2_RST	Independent watchdog 2 reset through EXTI.	-	35
6	38	IWDG4_RST	Independent watchdog 4 reset through EXTI.	-	37
8	39 to 40	-	-	-	-
9	41	WWDG1_RST	Window watchdog 1 reset through EXTI.	67	-
10	42	TAMP	Tamper interrupt (include LSECSS interrupts).	-	(16)
11	43	RTC	RTC global interrupt.	-	(17)
12	44	TAMP_S	Tamper secure interrupt (include LSECSS interrupts).	-	(21)
13	45	RTC_S	RTC global secure interrupt.	-	(22)
14	46	RCC	RCC global interrupt.	-	-
15	47	EXT2_0	EXT2 line 0 interrupt.	-	0
16	48	EXT2_1	EXT2 line 1 interrupt.	-	1

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
17	49	EXT2_2	EXT2 line 2 interrupt.	-	2
18	50	EXT2_3	EXT2 line 3 interrupt.	-	3
19	51	EXT2_4	EXT2 line 4 interrupt.	-	4
20	52	EXT2_5	EXT2 line 5 interrupt.	-	5
21	53	EXT2_6	EXT2 line 6 interrupt.	-	6
22	54	EXT2_7	EXT2 line 7 interrupt.	-	7
23	55	EXT2_8	EXT2 line 8 interrupt.	-	8
24	56	EXT2_9	EXT2 line 9 interrupt.	-	9
25	57	EXT2_10	EXT2 line 10 interrupt.	-	10
26	58	EXT2_11	EXT2 line 11 interrupt.	-	11
27	59	EXT2_12	EXT2 line 12 interrupt.	-	12
28	60	EXT2_13	EXT2 line 13 interrupt.	-	13
29	61	EXT2_14	EXT2 line 14 interrupt.	-	14
30	62	EXT2_15	EXT2 line 15 interrupt.	-	15
31	63	HPDMA1_Channel0	HPDMA1 channel0 interrupt.	-	-
32	64	HPDMA1_Channel1	HPDMA1 channel1 interrupt.	-	-
33	65	HPDMA1_Channel2	HPDMA1 channel2 interrupt.	-	-
34	66	HPDMA1_Channel3	HPDMA1 channel3 interrupt.	-	-
35	67	HPDMA1_Channel4	HPDMA1 channel4 interrupt.	-	-
36	68	HPDMA1_Channel5	HPDMA1 channel5 interrupt.	-	-
37	69	HPDMA1_Channel6	HPDMA1 channel6 interrupt.	-	-
38	70	HPDMA1_Channel7	HPDMA1 channel7 interrupt.	-	-
39	71	HPDMA1_Channel8	HPDMA1 channel8 interrupt.	-	-
40	72	HPDMA1_Channel9	HPDMA1 channel9 interrupt.	-	-
41	73	HPDMA1_Channel10	HPDMA1 channel10 interrupt.	-	-
42	74	HPDMA1_Channel11	HPDMA1 channel11 interrupt.	-	-
43	75	HPDMA1_Channel12	HPDMA1 channel12 interrupt.	-	-
44	76	HPDMA1_Channel13	HPDMA1 channel13 interrupt.	-	-
45	77	HPDMA1_Channel14	HPDMA1 channel14 interrupt.	-	-
46	78	HPDMA1_Channel15	HPDMA1 channel15 interrupt.	-	-
47	79	HPDMA2_Channel0	HPDMA2 channel0 interrupt.	-	-
48	80	HPDMA2_Channel1	HPDMA2 channel1 interrupt.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
49	81	HPDMA2_Channel2	HPDMA2 channel2 interrupt.	-	-
50	82	HPDMA2_Channel3	HPDMA2 channel3 interrupt.	-	-
51	83	HPDMA2_Channel4	HPDMA2 channel4 interrupt.	-	-
52	84	HPDMA2_Channel5	HPDMA2 channel5 interrupt.	-	-
53	85	HPDMA2_Channel6	HPDMA2 channel6 interrupt.	-	-
54	86	HPDMA2_Channel7	HPDMA2 channel7 interrupt.	-	-
55	87	HPDMA2_Channel8	HPDMA2 channel8 interrupt.	-	-
56	88	HPDMA2_Channel9	HPDMA2 channel9 interrupt.	-	-
57	89	HPDMA2_Channel10	HPDMA2 channel10 interrupt.	-	-
58	90	HPDMA2_Channel11	HPDMA2 channel11 interrupt.	-	-
59	91	HPDMA2_Channel12	HPDMA2 channel12 interrupt.	-	-
60	92	HPDMA2_Channel13	HPDMA2 channel13 interrupt.	-	-
61	93	HPDMA2_Channel14	HPDMA2 channel14 interrupt.	-	-
62	94	HPDMA2_Channel15	HPDMA2 channel15 interrupt.	-	-
63	95	HPDMA3_Channel0	HPDMA3 channel0 interrupt.	-	-
64	96	HPDMA3_Channel1	HPDMA3 channel1 interrupt.	-	-
65	97	HPDMA3_Channel2	HPDMA3 channel2 interrupt.	-	-
66	98	HPDMA3_Channel3	HPDMA3 channel3 interrupt.	-	-
67	99	HPDMA3_Channel4	HPDMA3 channel4 interrupt.	-	-
68	100	HPDMA3_Channel5	HPDMA3 channel5 interrupt.	-	-
69	101	HPDMA3_Channel6	HPDMA3 channel6 interrupt.	-	-
70	102	HPDMA3_Channel7	HPDMA3 channel7 interrupt.	-	-
71	103	HPDMA3_Channel8	HPDMA3 channel8 interrupt.	-	-
72	104	HPDMA3_Channel9	HPDMA3 channel9 interrupt.	-	-
73	105	HPDMA3_Channel10	HPDMA3 channel10 interrupt.	-	-
74	106	HPDMA3_Channel11	HPDMA3 channel11 interrupt.	-	-
75	107	HPDMA3_Channel12	HPDMA3 channel12 interrupt.	-	-
76	108	HPDMA3_Channel13	HPDMA3 channel13 interrupt.	-	-
77	109	HPDMA3_Channel14	HPDMA3 channel14 interrupt.	-	-
78	110	HPDMA3_Channel15	HPDMA3 channel15 interrupt.	-	-
79	111	ICACHE	ICACHE interrupt.	-	-
80	112	DCACHE	DCACHE interrupt.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
81	113	ADC1	ADC1 interrupt.	-	-
82	114	ADC2	ADC2 interrupt.	-	-
83	115	FDCAN_CAL	FDCAN CCU interrupt.	-	-
84	116	FDCAN1_IT0	FDCAN1 interrupt 0.	-	-
85	117	FDCAN2_IT0	FDCAN2 interrupt 0.	-	-
86	118	FDCAN1_IT1	FDCAN1 interrupt 1.	-	-
87	119	FDCAN2_IT1	FDCAN2 interrupt 1.	-	-
88	120	TIM1_BRK	TIM1 Break interrupt.	-	-
89	121	TIM1_UP	TIM1 Update interrupt.	-	-
90	122	TIM1_TRG_COM	TIM1 trigger and commutation interrupts.	-	-
91	123	TIM1_CC	TIM1 capture compare interrupt.	-	-
92	124	TIM2	TIM2 global interrupt.	-	-
93	125	TIM3	TIM3 global interrupt.	-	-
94	126	TIM4	TIM4 global interrupt.	-	-
95	127	I2C1	I2C1 global interrupt.	(21)	-
96	128	I3C1	I3C1 global interrupt.	(74)	-
97	129	I2C2	I2C2 global interrupt.	(22)	-
98	130	I3C2	I3C2 global interrupt.	(75)	-
99	131	SPI1	SPI1 global interrupt.	(36)	-
100	132	SPI2	SPI2 global interrupt.	(37)	-
101	133	USART1	USART1 global interrupt.	(26)	-
102	134	USART2	USART2 global interrupt.	(27)	-
103	135	USART3	USART3 global interrupt.	(28)	-
104	136	-	Reserved.	-	-
105	137	TIM8_BRK	TIM8 break interrupt.	-	-
106	138	TIM8_UP	TIM8 update interrupt.	-	-
107	139	TIM8_TRG_COM	TIM8 trigger & commutation interrupt.	-	-
108	140	TIM8_CC	TIM8 capture compare interrupt.	-	-
109	141	FMC	FMC global interrupt.	-	-
110	142	SDMMC1	SDMMC1 global interrupt.	-	-
111	143	TIM5	TIM5 global interrupt.	-	-
112	144	SPI3	SPI3 global interrupt.	(38)	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
113	145	UART4	UART4 global interrupt.	(30)	-
114	146	UART5	UART5 global interrupt.	(31)	-
115	147	TIM6	TIM6 global interrupt.	-	-
116	148	TIM7	TIM7 global interrupt.	-	-
117	149	ETH1_SBD	ETH1 global interrupt.	-	-
118	150	ETH1_PMT	ETH1 wakeup interrupt (PMT).	(68)	-
119	151	ETH1_LPI	ETH1 LPI interrupt.	-	-
120	152	ETH2_SBD	ETH2 global interrupt.	-	-
121	153	ETH2_PMT	ETH2 wakeup interrupt (PMT).	(70)	-
122	154	ETH2_LPI	ETH2 LPI interrupt.	-	-
123	155	USART6	USART6 global interrupt.	(29)	-
124	156	I2C3	I2C3 global interrupt.	(23)	-
125	157	I3C3	I3C3 global interrupt.	(76)	-
126	158	USBH_EHCI	USB host EHCI interrupt.	-	-
127	159	USBH_OHCI	USB host OHCI interrupt.	-	-
128	160	DCMI_PSSI	DCMI & PSSI global interrupt.	-	-
129	161	CSI2HOST	CSI2 host controller interrupt.	-	-
130	162	-	Reserved.	-	-
131	163	CRYP1	Crypto1 interrupt.	-	-
132	164	HASH1	Hash1 interrupt.	-	-
133	165	PKA	PKA interrupt.	-	-
134	166	-	-	-	-
135	167	UART7	UART7 global interrupt.	(32)	-
136	168	LPUART1	LPUART1 global interrupt.	-	(26)
137	169	SPI4	SPI4 global interrupt.	(39)	-
138	170	SPI5	SPI5 global interrupt.	(40)	-
139	171	SPI6	SPI6 global interrupt.	(41)	-
140	172	SAI1	SAI1 global interrupt.	-	-
141	173	LTDC_IRQ	LTDC global interrupt.	-	-
142	174	LTDC_ER_IRQ	LTDC global error interrupt.	-	-
143	175	LTDC_SEC_IRQ	LTDC security global interrupt.	-	-
144	176	LTDC_SEC_ER_IRQ	LTDC security global error interrupt.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
145	177	SAI2	SAI2 global interrupt.	-	-
146	178	OCTOSPI1	OCTOSPI1 global interrupt.	-	-
147	179	-	Reserved.	-	-
148	180	OTFDEC1	OTFDEC1 interrupt.	-	-
149	181	LPTIM1	LPTIMER1 global interrupt.	(47)	-
150	182	USBH WAKEUP	USBH remote wakeup from USB2PHY1.	43	-
151	183	SPDIFRX	SPDIFRX global interrupt.	-	-
152	184	IPCC1_RX	Mailbox 1 RX occupied interrupt.	(59)	-
153	185	IPCC1_TX	Mailbox 1 TX free interrupt.	(59)	-
154	186	IPCC1_RX_S	Mailbox 1 RX occupied secure interrupt.	(61)	-
155	187	IPCC1_TX_S	Mailbox 1 TX free secure interrupt.	(61)	-
156	188	SAES	Secure AES.	-	-
157	189	CRYP2	Crypto2 interrupt.	-	-
158	190	OTG WAKEUP	USB2 DR remote wakeup from USB2PHY2.	44	-
159	191	-	Reserved.	-	-
160	192	MDF1_FLT0	MDF1 filter0 interrupt.	-	-
161	193	MDF1_FLT1	MDF1 filter1 interrupt.	-	-
162	194	MDF1_FLT2	MDF1 filter2 interrupt.	-	-
163	195	MDF1_FLT3	MDF1 filter3 interrupt.	-	-
164	196	SAI3	SAI3 global interrupt.	-	-
165	197	TIM15	TIM15 global interrupt.	-	-
166	198	TIM16	TIM16 global interrupt.	-	-
167	199	TIM17	TIM17 global interrupt.	-	-
168	200	TIM12	TIM12 global interrupt.	-	-
169	201	SDMMC2	SDMMC2 global interrupt.	-	-
170	202	DCMIPP	DCMIPP global interrupt.	-	-
171	203	-	-	-	-
172	204	CPU2_SYSRESETQ	MCU local reset request through EXTI.	73	-
173	205	TIM13	TIM13 global interrupt.	-	-
174	206	TIM14	TIM14 global interrupt.	-	-
175	207	TIM10	TIM10 global interrupt.	-	-
176	208	RNG1	RNG1 interrupt.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
177	209	-	-.	-	-
178	210	HASH2	HASH2 interrupt.	-	-
179	211	SDMMC3	SDMMC3 global interrupt.	-	-
180	212	LPTIM2	LPTIMER2 global interrupt.	(48)	-
181	213	LPTIM3	LPTIMER3 global interrupt.	-	(29)
182	214	LPTIM4	LPTIMER4 global interrupt.	-	(30)
183	215	LPTIM5	LPTIMER5 global interrupt.	-	(31)
184	216	-	Reserved.	-	-
185	217	CPU2_SEV	Cortex-M33 send event through EXTI.	64	-
186	218	RCC_WAKEUP	RCC CPU1 wakeup interrupt.	-	-
187	219	SAI4	SAI4 global interrupt.	-	-
188	220	DTS	Temperature sensor interrupt.	(72)	-
189	221	TIM11	TIM11 global interrupt.	-	-
190	222	CPU1_WAKEUP_PIN	Interrupt for all 6 wakeup enabled by CPU1.	(52, 53, 54, 55, 56, 57)	-
191	223	OTG	USB2 DR global interrupt.	-	-
192	224	RNG2	RNG2 interrupt.	-	-
193	225	-	Reserved.	(46)	-
194	226	Rserved	Reserved.	-	-
195	227	SERF	SERF global interrupt.	-	-
196	228	-	Reserved.	-	-
197	229	RAMCFG	RAM configuration global interrupt.	-	-
198	230	DDRCTRL	DDRCTRL interrupt.	-	-
199	231	DDRPHYC	DDRPHY interrupt (active low).	-	-
200	232	DFI_ERR	DDRPHY DFI error interrupt.	-	-
201	233	IAC	RIF illegal access controller interrupt.	-	(49)
202	234	VDDCPU_VD	VDDCPU voltage detector.	-	51
203	235	VDDCORE_VD	VDDCORE voltage detector.	-	52
204	236	RETRAMCRC_ERR_IT	RETRAM CRC error.	-	53

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
205	237	HPDMA1 WKUP	HPDMA1 channel 0 to 15 wakeup.	(77)	-
206	238	HPDMA2 WKUP	HPDMA2 channel 0 to 15 wakeup.	(78)	-
207	239	HPDMA3 WKUP	HPDMA3 channel 0 to 15 wakeup.	(79)	-
208	240	DBG_AUTH_HOST	Debugger has written DBG_AUTH_HOST register.	-	-
209	241	DBG_AUTH_ACK	Debugger has read DBG_AUTH_ACK register.	-	-
210	242	RCC_HSI_FMON	HSI frequency monitor.	(19)	-
211	243	RCC_MSI_FMON	MSI frequency monitor.	(18)	-
212 to 267	244 to 299	-	-	-	-
268	300	EXTI1_0	EXTI1 Line 0 interrupt	0	-
269	301	EXTI1_1	EXTI1 Line 1 interrupt	1	-
270	302	EXTI1_2	EXTI1 Line 2 interrupt	2	-
271	303	EXTI1_3	EXTI1 Line 3 interrupt	3	-
272	304	EXTI1_4	EXTI1 Line 4 interrupt	4	-
273	305	EXTI1_5	EXTI1 Line 5 interrupt	5	-
274	306	EXTI1_6	EXTI1 Line 6 interrupt	6	-
275	307	EXTI1_7	EXTI1 Line 7 interrupt	7	-
276	308	EXTI1_8	EXTI1 Line 8 interrupt	8	-
277	309	EXTI1_9	EXTI1 Line 9 interrupt	9	-
278	310	EXTI1_10	EXTI1 Line 10 interrupt	10	-
279	311	EXTI1_11	EXTI1 Line 11 interrupt	11	-
280	312	EXTI1_12	EXTI1 Line 12 interrupt	12	-
281	313	EXTI1_13	EXTI1 Line 13 interrupt	13	-
282	314	EXTI1_14	EXTI1 Line 14 interrupt	14	-
283	315	EXTI1_15	EXTI1 Line 15 interrupt	15	-
284 to 307	316 to 339	-	-	-	-
308	340	-	Reserved.	-	-
309	341	L2WERRIRQ	Error indicator from L2 for memory transactions with a write response error condition.	-	-
310	342	DDRPERFM	DDR performance monitor interrupt.	-	-
311	343	-	Reserved.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
312	344	PMSTAT0	Power management, PLL and sensor interrupts.	-	-
313	345	PMSTAT1	Power management, PLL and sensor interrupts.	-	-
314	346	PMSTAT2	Power management, PLL and sensor interrupts.	-	-
315	347	PMSTAT3	Power management, PLL and sensor interrupts.	-	-
316	348	PMSTAT4	Power management, PLL and sensor interrupts.	-	-
317	349	PMSTAT5	Power management, PLL and sensor interrupts.	-	-
318	350	PMSTAT6	Power management, PLL and sensor interrupts.	-	-
319	351	PMSTAT7	Power management, PLL and sensor interrupts.	-	-
320	352	PMSTAT8	Power management, PLL and sensor interrupts.	-	-
321	353	PMSTAT9	Power management, PLL and sensor interrupts.	-	-
322	354	PMSTAT10	Power management, PLL and sensor interrupts.	-	-
323	355	PMSTAT11	Power management, PLL and sensor interrupts.	-	-
324	356	PMSTAT12	Power management, PLL and sensor interrupts.	-	-
325	357	PMSTAT13	Power management, PLL and sensor interrupts.	-	-
326	358	PMSTAT14	Power management, PLL and sensor interrupts.	-	-
327	359	PMSTAT15	Power management, PLL and sensor interrupts.	-	-
328	360	PMSTAT16	Power management, PLL and sensor interrupts.	-	-
329 to 367	361 to 399	-	Reserved.	-	-
368	400	PMUIRQ0	Cortex-A35 Core#0 performance monitor interrupt.	-	-
369	401	PMUIRQ1	Cortex-A35 Core#1 performance monitor interrupt.	-	-
370 to 371	402 to 403	-	Reserved.	-	-
372	404	COMMRX0	Cortex-A35 Core#0 debug communication channel receive interrupt.	-	-
373	405	COMMRX1	Cortex-A35 Core#1 debug communication channel receive interrupt.	-	-
374	406	-	Reserved.	-	-
375	407	-	Reserved.	-	-
376	408	COMMTX0	Cortex-A35 Core#0 debug communication channel transmit interrupt.	-	-
377	409	COMMTX1	Cortex-A35 Core#1 debug communication channel transmit interrupt.	-	-

Table 188. STM32MP21xx Interrupt Mapping for Cortex-A35 (continued)

Number	ID	Acronym	Description	EXTI1 event	EXTI2 event
378	410	-	Reserved.	-	-
379	411	-	Reserved.	-	-
380	412	nCTIIRQ0	Cortex-A35 Core#0 CTI interrupt.	-	-
381	413	nCTIIRQ1	Cortex-A35 Core#1 CTI interrupt.	-	-
382	414	-	Reserved.	-	-
383	415	-	Reserved.	-	-

27 Nested vectored interrupt controller (NVIC)

27.1 NVIC features

The NVIC for Cortex-M33 (CPU2) includes the following features:

- 320 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the Cortex-M33 core are closely coupled, which enables low-latency interrupt processing, and efficient processing of late arriving interrupts.

The NVIC registers are banked across secure and non-secure states.

All the interrupts including Cortex-M33 core exceptions are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to *STM32 Cortex-M33 MCUs programming manual* (PM0264).

27.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers, that are available when TrustZone is activated:

- SysTick secure instance
- SysTick non-secure instance

When TrustZone is disabled, only one SysTick timer is available.

As SysTick for Cortex-M33 does not run on a fixed frequency, there is no SysTick calibration value available.

The SysTick reload counter must be set according to the required time base, and the ck_cpu2 frequency value must be set in the RCC.

27.3 Interrupt and exception vectors

See Cortex-M33 interrupt table in *Section 26: Interrupt list*.

28 Global interrupt controller (GIC)

28.1 GIC introduction

The following Arm specifications provide additional details on the GIC:

- Arm generic interrupt controller - Architecture specification v2.0
- Cortex-A35 technical reference manual revision r0p1

28.2 GIC main features

The integrated GIC collates and arbitrates from a large number of interrupt sources. It provides:

- Masking of interrupts
- Prioritization of interrupts
- Distribution of the interrupts to the target processors
- Tracking of the status of interrupts
- Generation of interrupts by software
- Support for security extensions
- Support for virtualization extensions

28.3 GIC functional description

The GIC is a single functional unit located in the Cortex-A35. It consists of a shared distributor block, and several CPU interfaces. Each Cortex-A35 processor core embeds:

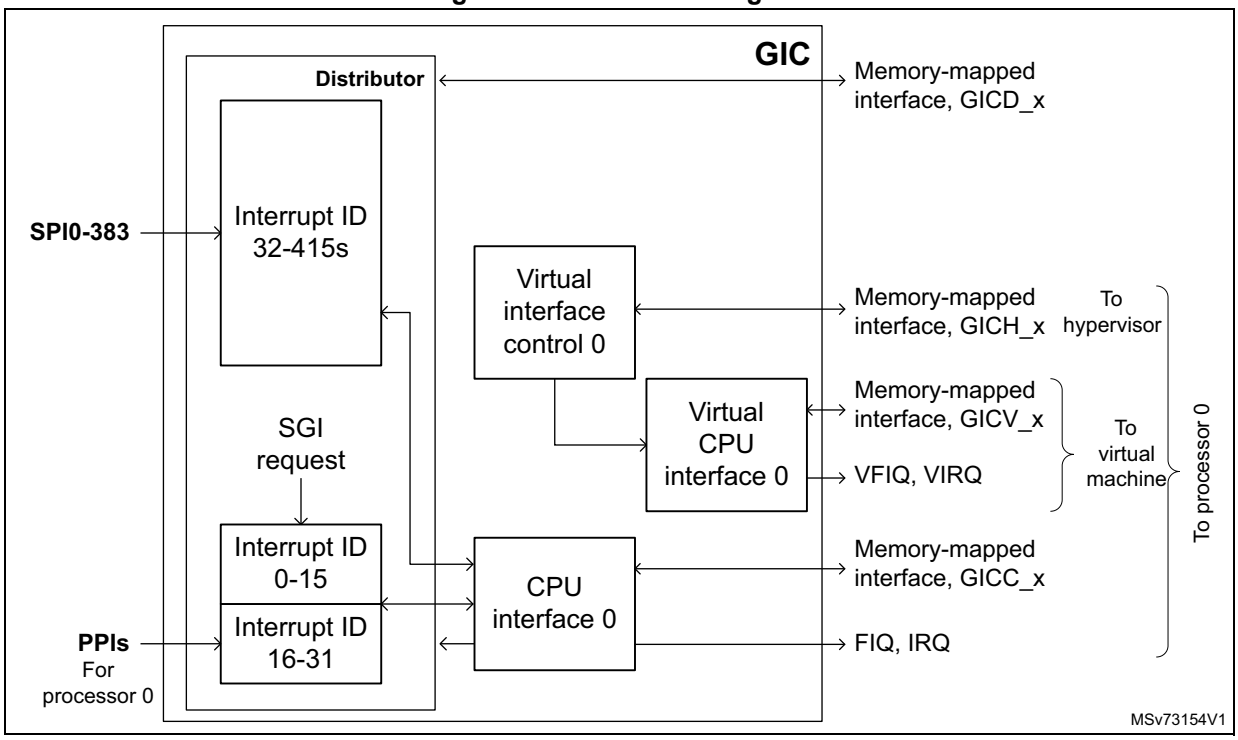
- a CPU interface
- a virtual interface control
- a virtual CPU interface

The GIC registers are memory-mapped, and the base address is visible into the configuration base address (CBAR) for each processor core in the cluster.

Memory regions used for these registers must be marked as device, or strongly-ordered in the translation tables.

Memory regions marked as normal memory cannot access any of the GIC registers: access caches or external memory as required.

Figure 157. GIC block diagram



28.4 GIC interrupt sources

All interrupt sources are identified by a unique ID. The Cortex-A35 processor interrupt sources are detailed below.

28.4.1 Software generated interrupts (SGI)

The SGIs are generated by writing to the software generated interrupt register (GICD_SGIR). Each processor interface can generate a maximum of 16 SGIs (ID0-ID15). An SGI has edge-triggered properties. The software triggering of the interrupt is equivalent to the edge transition of the interrupt signal on a peripheral input.

Note: Arm recommends to use SGI ID0-ID7 for non-secure interrupts, and SGI ID8-ID15 for secure interrupts.

28.4.2 Private peripheral interrupts (PPI)

A PPI is an interrupt generated by a peripheral that is specific to a single processor. There are six PPIs for each CPU interface:

- legacy nFIQ signal (PPI0) - not used

  When the GIC interrupt bypass is in effect (such as after reset), the external nFIQ signal bypasses the interrupt distributor logic, and directly drives the interrupt request to the corresponding processor.

  When a processor uses the GIC rather than the external nFIQ signal (by enabling its own CPU interface), the nFIQ signal is treated like other interrupt lines, and uses ID28. The interrupt is active-low-level sensitive.

- secure physical timer event (PPI1)

  This event is generated from the secure physical timer, and uses ID29. The interrupt is level sensitive.
- non-secure physical timer event (PPI2)

  This event is generated from the non-secure physical timer, and uses ID30. The interrupt is level sensitive.
- legacy nIRQ signal (PPI3) - not used

  When the GIC interrupt bypass is in effect (such as after reset), the external nIRQ signal bypasses the interrupt distributor logic, and directly drives the interrupt request to the corresponding processor.

  When a processor uses the GIC rather than the external nIRQ signal (by enabling its own CPU interface), the nIRQ signal is treated like other interrupt lines and uses ID31. The interrupt is active-low-level sensitive.
- virtual timer event (PPI4)

  This event is generated from the virtual timer, and uses ID27. The interrupt is level sensitive.
- hypervisor timer event (PPI5)

  This event is generated from the physical timer in hypervisor mode, and uses ID26. The interrupt is level-sensitive.

28.4.3 Shared peripheral interrupts (SPI)

The SPIs are triggered by events generated on associated interrupt input lines. The GIC supports up to 384 SPIs corresponding to the external IRQS signal.

The SPIs start at ID32, and can be configured to be edge-triggered or active-high-level sensitive.

28.4.4 Interrupt priority formats

The Cortex-A35 implements a 5-bit version of the interrupt priority field for 32 interrupt priority levels.

28.5 GIC distributor (GICD)

The GICD centralizes all interrupt sources, determines the priority of each interrupt, and, for each CPU interface, forwards the interrupt with the highest priority to the interface for priority masking and preemption handling.

The GICD provides a programming interface for the following actions:

- Enable globally the forwarding of interrupts to the CPU interfaces.
- Enable/disable each interrupt.
- Set the priority level of each interrupt.
- Set the target processor list of each interrupt.
- Set each peripheral interrupt to be level sensitive or edge triggered.
- Set each interrupt as either group 0 or group 1.
- Send an SGI to one or more target processor cores.
- Give visibility of the state of each interrupt.

- Provide a mechanism for software to set/clear pending state of a peripheral interrupt.

The GICD_IPRIORITYRx, GICD_ITARGETSRx, GICD_CPENDSGIRx, and GICD_SPENDSGIRx registers are byte-accessible and halfword-accessible.

All other registers are word-accessible.

28.5.1 GICD control register (GICD_CTLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLEGRP1	ENABLEGRP0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ENABLEGRP1:** Enable group 1 interrupts
Global enable for forwarding pending group 1 interrupts from the GICD to the CPU interfaces

Bit 0 **ENABLEGRP0**: Enable group 0 interrupts
Global enable for forwarding pending group 0 interrupts from the GICD to the CPU interfaces

28.5.2 GICD control non-secure access register (GICD_CTLRNS)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
Global enable for forwarding pending group 1 interrupts from the GICD to the CPU interfaces

28.5.3 GICD interrupt controller type register (GICD_TYPER)

Address offset: 0x004

Reset value: 0x0000 FC0C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSPI[4:0]					SECURITYEXTN	Res.	Res.	CPUNUMBER[2:0]			ITLINESNUMBER[4:0]				
r	r	r	r	r	r			r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **LSPI[4:0]**: Lockable shared peripheral interrupt
This bitfield returns the number of LSPIs that the interrupt controller contains. (0b1111 = 31 LSPIs, IDs 32-62).

Bit 10 **SECURITYEXTN**: Security extension
This bit indicates whether the GIC implements the security extensions. This bit always returns a value of 1, indicating that the security extensions are implemented.

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:5 **CPUNUMBER[2:0]**: Number of processor interfaces
This bitfield indicates the number of implemented processor interfaces in the GIC. (0b000 = 1 processor interface)

Bits 4:0 **ITLINESNUMBER[4:0]**: Number of interrupt lines
This bitfield indicates the number of interrupts that the GIC supports. (0b01100= up to 416 interrupts, up to 384 SPI interrupts)

28.5.4 GICD implementer identification register (GICD_IIDR)

Address offset: 0x008

Reset value: 0x0202 143B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCTID[7:0]								Res.	Res.	Res.	Res.	VARIANT[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION[3:0]				IMPLEMENTER[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PRODUCTID[7:0]**: GIC product ID

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **VARIANT[3:0]**: GIC major revision number

Bits 15:12 **REVISION[3:0]**: GIC minor revision number

Bits 11:0 **IMPLEMENTER[11:0]**: GIC implementer (0x43B Arm implementation)

28.5.5 GICD interrupt group register x (GICD_IGROUPRx)

Address offset: 0x080 + 0x4 * x, (x = 0 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IGROUPRx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IGROUPRx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IGROUPRx[31:0]**: Group x of interrupts

28.5.6 GICD interrupt set-enable register (GICD_ISENABLER0)

Address offset: 0x100

Reset value: 0b0000 00XX XXXX XXXX 1111 1111 1111 1111

For interrupts, ID = 0 to IFD = 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISENABLER0[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISENABLER0[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISENABLER0[31:0]**: Interrupt set-enable 0

28.5.7 GICD interrupt set-enable register x (GICD_ISENABLERx)

Address offset: 0x104 + 0x4 * (x - 1), (x = 1 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISENABLERx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISENABLERx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISENABLERx[31:0]:** Interrupt set-enable x

28.5.8 GICD interrupt clear-enable register (GICD_ICENABLER0)

Address offset: 0x180

Reset value: 0b0000 00XX XXXX XXXX 1111 1111 1111 1111

For interrupts, ID = 0 to ID = 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICENABLER0[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICENABLER0[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICENABLER0[31:0]**: Interrupt clear-enable 0

28.5.9 GICD interrupt clear-enable register x (GICD_ICENABLERx)

Address offset: 0x184 + 0x4 * (x - 1), (x = 1 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICENABLERx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICENABLERx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICENABLERx[31:0]**: Interrupt clear-enable x

28.5.10 GICD interrupt set-pending register x (GICD_ISPENDRx)

Address offset: 0x200 + 0x4 * x, (x = 0 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISPENDRx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISPENDRx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISPENDRx[31:0]**: Interrupt set-pending x

28.5.11 GICD interrupt clear-pending register x (GICD_ICPENDRx)

Address offset: 0x280 + 0x4 * x, (x = 0 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICPENDRx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICPENDRx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICPENDRx[31:0]**: Interrupt clear-pending x

28.5.12 GICD interrupt set-active register x (GICD_ISACTIVERx)

Address offset: 0x300 + 0x4 * x, (x = 0 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISACTIVERx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISACTIVERx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISACTIVERx[31:0]**: Interrupt set-active x

28.5.13 GICD interrupt clear-active register x (GICD_ICACTIVERx)

Address offset: 0x380 + 0x4 * x, (x = 0 to 12)

Reset value: 0x0000 0000

For interrupts, ID = x * 32 to ID = x * 32 + 31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICACTIVERx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICACTIVERx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICACTIVERx[31:0]**: Interrupt clear-active x

28.5.14 GICD interrupt priority register x (GICD_IPRIORITYRx)

Address offset: 0x400 + 0x4 * x, (x = 0 to 103)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIORITY3[4:0]					Res.	Res.	Res.	PRIORITY2[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIORITY1[4:0]					Res.	Res.	Res.	PRIORITY0[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw			

Bits 31:27 **PRIORITY3[4:0]:** Priority for interrupt ID = x * 4 + 3

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:19 **PRIORITY2[4:0]:** Priority for interrupt ID = x * 4 + 2

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:11 **PRIORITY1[4:0]:** Priority for interrupt ID = x * 4 + 1

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY0[4:0]:** Priority for interrupt ID = x * 4

Bits 2:0 Reserved, must be kept at reset value.

28.5.15 GICD interrupt processor target register x (GICD_ITARGETSRx)

Address offset: 0x800 + 0x4 * x, (x = 0 to 7)

Reset value: 0xXXXX XXXX

For existing SGIs and PPIs, a read of CPU targets field returns the number of the processor performing the read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 2[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 0[1:0]	
						r	r							r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **CPU_TARGETS3[1:0]**: CPU target for interrupt ID = x * 4 + 3

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPU_TARGETS2[1:0]**: CPU target for interrupt ID = x * 4 + 2

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CPU_TARGETS1[1:0]**: CPU target for interrupt ID = x * 4 + 1

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **CPU_TARGETS0[1:0]**: CPU target for interrupt ID = x * 4

28.5.16 GICD interrupt processor target register x (GICD_ITARGETSRx)

Address offset: 0x820 + 0x4 * (x - 8), (x = 8 to 103)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 2[1:0]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS 0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **CPU_TARGETS3[1:0]:** CPU target for interrupt ID = x * 4 + 3

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPU_TARGETS2[1:0]:** CPU target for interrupt ID = x * 4 + 2

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CPU_TARGETS1[1:0]**: CPU target for interrupt ID = x * 4 + 1

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **CPU_TARGETS0[1:0]**: CPU target for interrupt ID = x * 4

28.5.17 GICD interrupt configuration register (GICD_ICFGR0)

Address offset: 0xC00

Reset value: 0xAAAA AAAA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INT_CONFIGy[1:0]**: Interrupt config for interrupt ID = y (y = 15 to 0)

28.5.18 GICD interrupt configuration register (GICD_ICFGR1)

Address offset: 0xC04

Reset value: 0b0101 0101 0101 01XX XXXX XXXX XXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INT_CONFIGy[1:0]**: Interrupt config for interrupt ID = y + 16 (y = 15 to 0)

28.5.19 GICD interrupt configuration register x (GICD_ICFGRx)

Address offset: 0xC08 + 0x4 * (x - 2), (x = 2 to 25)

Reset value: 0x5555 5555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INT_CONFIGy[1:0]**: Interrupt config for interrupt ID = x * 16 + y (y = 15 to 0)

28.5.20 GICD private peripheral interrupt status register (GICD_PPISR)

Address offset: 0xD00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPI3	PPI2	PPI1	PPI0	PPI4	PPI5	PPI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r									

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PPI3**: nIRQ (not used)

Bit 14 **PPI2**: Secure physical timer event

Bit 13 **PPI1**: Secure physical timer event

Bit 12 **PPI0**: nFIQ (not used)

Bit 11 **PPI4**: Virtual timer event

Bit 10 **PPI5**: Hypervisor timer event

Bit 9 **PPI6**: Virtual maintenance interrupt

Bits 8:0 Reserved, must be kept at reset value.

28.5.21 GICD shared peripheral interrupt register x (GICD_SPISRx)

Address offset: 0xD04 + 0x4 * x, (x = 0 to 11)

Reset value: 0x0000 0000

For interrupts, ID = SPI number + 32, from SPI = x * 32 + 31 to SPI = x * 32.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPISRx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPISRx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SPISRx[31:0]**: Shared peripheral x interrupt

28.5.22 GICD software generated interrupt register (GICD_SGIR)

Address offset: 0xF00

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TARGETLISTFILTER[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPUTARGETLIST[1:0]	
						w	w							w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSATT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SGIINTID[3:0]			
w												w	w	w	w

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **TARGETLISTFILTER[1:0]**: Target list filter

This bitfield determines how the distributor must process the requested SGI.

0: Forward the interrupt to the CPU interfaces specified in the CPUTARGETLIST bitfield.

1: Forward the interrupt to all CPU interfaces except the processor which requested the interrupt.

2: Forward the interrupt only to the CPU interface of the processor which requested the interrupt.

3: Reserved

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPUTARGETLIST[1:0]**: CPU target list

When TARGETLISTFILTER = 0b00, this CPUTARGETLIST bitfield defines the CPU interfaces to which the GICD must forward the interrupt. Each bit refers to the corresponding CPU interface. If this bitfield is 0 when TARGETLISTFILTER = 0b00, the GICD does not forward the interrupt to any CPU interface.

Bit 15 **NSATT**: Non-secure attribute

this bit specifies the required security value of the SGI.

0: Forward the SGI specified in the SGIINTID bitfield to a specified CPU interface, only if the SGI is configured as group 0 on this interface.

1: Forward the SGI specified in the SGIINTID bitfield to a specified CPU interface, only if the SGI is configured as group 1 on this interface.

Note: This bit is writable only by a secure access. Any non-secure write to this register generates an SGI, only if the specified SGI is programmed as group 1.

Bits 14:4 Reserved, must be kept at reset value.

Bits 3:0 **SGIINTID[3:0]**: SGI interrupt ID

The value of this bitfield is the interrupt ID of the SGI to forward to the specified CPU interfaces, in the range 0-15 (for example: a value of 0b0011 specifies interrupt ID 3).

28.5.23 GICD SGI clear-pending register x (GICD_CPENDSGIRx)

Address offset: 0xF10 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

For SGI = x * 4 to SGI = x * 4 + 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING2[1:0]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **SGI_CLEAR_PENDING3[1:0]**: Clear-pending state for SGI = x * 4 + 3

Writing a 1 clears the pending state of the SGI for the corresponding source processor, and no longer targets the processor performing the write.

Writing a 0 has no effect.

Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **SGI_CLEAR_PENDING2[1:0]**: Clear-pending state for SGI = x * 4 + 2

Writing a 1 clears the pending state of the SGI for the corresponding source processor, and no longer targets the processor performing the write.

Writing a 0 has no effect.

Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **SGI_CLEAR_PENDING1[1:0]**:Clear-pending state for SGI = x * 4 + 1

Writing a 1 clears the pending state of the SGI for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **SGI_CLEAR_PENDING0[1:0]**: Clear-pending state for SGI = x * 4

Writing a 1 clears the pending state of the SGI for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

28.5.24 GICD SGI set-pending register x (GICD_SPENDSGIRx)

Address offset: 0xF20 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

For SGI = x * 4 to SGI = x * 4 + 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING2[1:0]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **SGI_SET_PENDING3[1:0]**: Set-pending state for SGI = x * 4 + 3

Writing a 1 sets the pending state of the SGI for the corresponding source processor. Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **SGI_SET_PENDING2[1:0]**: Set-pending state for SGI= x * 4 + 2

Writing a 1 sets the pending state of the SGI for the corresponding source processor. Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **SGI_SET_PENDING1[1:0]**: Set-pending state for SGI = x * 4 + 1

Writing a 1 sets the pending state of the SGI for the corresponding source processor. Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **SGI_SET_PENDING0[1:0]**: Set-pending state for SGI = x * 4
Writing a 1 sets the pending state of the SGI for the corresponding source processor.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI is pending, from the corresponding source processor, on the reading processor.

28.5.25 GICD peripheral ID4 register (GICD_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR4[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR4[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR4[31:0]**: Peripheral ID4

28.5.26 GICD peripheral ID5 to ID7 register x (GICD_PIDRx)

Address offset: 0xFD4 + 0x4* (x - 5), (x = 5 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDRx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDRx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDRx[31:0]**: Peripheral ID x

28.5.27 GICD peripheral ID0 register (GICD_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR0[31:0]**: Peripheral ID0

28.5.28 GICD peripheral ID1 register (GICD_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR1[31:0]**: Peripheral ID1

28.5.29 GICD peripheral ID2 register (GICD_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR2[31:0]**: Peripheral ID2

28.5.30 GICD peripheral ID3 register (GICD_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR3[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR3[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR3[31:0]**: Peripheral ID3

28.5.31 GICD component ID0 register (GICD_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR0[31:0]**: Component ID0

28.5.32 GICD component ID1 register (GICD_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR1[31:0]**: Component ID1

28.5.33 GICD component ID2 register (GICD_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR2[31:0]**: Component ID2

28.5.34 GICD component ID3 register (GICD_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR3[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR3[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR3[31:0]**: Component ID3

28.5.35 GICD register map

Table 189. GICD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICD_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLEGRP1	ENABLEGRP0
	Reset value																															0	0
0x000	GICD_CTLRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0x004	GICD_TYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSPI[4:0]					SECURITYEXTN	Res.	Res.	CPUNUMBER[2:0]			ITLINES NUMBER[4:0]				
	Reset value																	1	1	1	1	1	1			0	0	0	0	1	1	0	0
0x008	GICD_IIDR	PRODUCTID[7:0]								Res.	Res.	Res.	Res.	VARIANT [3:0]				REVISION [3:0]				IMPLEMENTER[11:0]											
	Reset value	0	0	0	0	0	0	1	0					0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x00C-0x07C	Reserved	Reserved																															
0x080 + 0x4*x (x=0 to 12) Last address: 0x0B0	GICD_IGROUPRx	IGROUPRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4-0x0FC	Reserved	Reserved																															
0x100	GICD_ ISENABLER0	ISENABLER0[31:0]																															
	Reset value	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x104 + 0x4*(x-1), (x=1 to 12) Last address: 0x130	GICD_ ISENABLERx	ISENABLERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134-0x17C	Reserved	Reserved																															
0x180	GICD_ ICENABLER0	ICENABLER0[31:0]																															
	Reset value	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 189. GICD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184 + 0x4*(x-1), (X=1 to 12) Last address: 0x1B0	GICD_ ICENABLERx	ICENABLERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B4 -0x1FC	Reserved	Reserved																															
0x200 + 0x4*x (x=0 to 12) Last address: 0x230	GICD_ISPENDRx	ISPENDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x234-0x27C	Reserved	Reserved																															
0x280 + 0x4*x (x=0 to 12) Last address: 0x2B0	GICD_ICPENDRx	ICPENDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2B4-0x2FC	Reserved	Reserved																															
0x300 + 0x4*x (x=0 to 12) Last address: 0x330	GICD_ ISACTIVERx	ISACTIVERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x334-0x37C	Reserved	Reserved																															
0x380 + 0x4*x (x=0 to 12) Last address: 0x3B0	GICD_ ICACTIVERx	ICACTIVERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B4-0x3FC	Reserved	Reserved																															
0x400 + 0x4*x (x=0 to 103) Last address: 0x59C	GICD_ IPRIORITYRx	PRIORITY3[4:0]				Res.	Res.	Res.		PRIORITY2[4:0]					Res.	Res.	Res.	PRIORITY1[4:0]					Res.	Res.	Res.	PRIORITY0[4:0]					Res.	Res.	Res.	
	Reset value	0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0			
0x5A0-0x7FC	Reserved	Reserved																															
0x800 + 0x4*x (x=0 to 7) Last address: 0x81C	GICD_ ITARGETSRx	Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0 [1:0]	
	Reset value							x	x							x	x							x	x							x	x
0x820 + 0x4*x (x=8 to 103) Last address: 0x99C	GICD_ ITARGETSRx	Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0 [1:0]	
	Reset value							0	0							0	0							0	0							0	0
0x9A0-0xBFC	Reserved	Reserved																															
0xC00	GICD_ICFGR0	INT_CONFIG15 [1:0]		INT_CONFIG14 [1:0]		INT_CONFIG13 [1:0]		INT_CONFIG12 [1:0]		INT_CONFIG11 [1:0]		INT_CONFIG10 [1:0]		INT_CONFIG9 [1:0]		INT_CONFIG8 [1:0]		INT_CONFIG7 [1:0]		INT_CONFIG6 [1:0]		INT_CONFIG5 [1:0]		INT_CONFIG4 [1:0]		INT_CONFIG3 [1:0]		INT_CONFIG2 [1:0]		INT_CONFIG1 [1:0]		INT_CONFIG0 [1:0]	
	Reset value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0xC04	GICD_ICFGR1	INT_CONFIG15 [1:0]		INT_CONFIG14 [1:0]		INT_CONFIG13 [1:0]		INT_CONFIG12 [1:0]		INT_CONFIG11 [1:0]		INT_CONFIG10 [1:0]		INT_CONFIG9 [1:0]		INT_CONFIG8 [1:0]		INT_CONFIG7 [1:0]		INT_CONFIG6 [1:0]		INT_CONFIG5 [1:0]		INT_CONFIG4 [1:0]		INT_CONFIG3 [1:0]		INT_CONFIG2 [1:0]		INT_CONFIG1 [1:0]		INT_CONFIG0 [1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 189. GICD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC08 + 0x4*(x-2), (x=2 to 25) Last address: 0xC64	GICD_ICFGRx	INT_CONFIG15 [1:0]		INT_CONFIG14 [1:0]		INT_CONFIG13 [1:0]		INT_CONFIG12 [1:0]		INT_CONFIG11 [1:0]		INT_CONFIG10 [1:0]		INT_CONFIG9 [1:0]		INT_CONFIG8 [1:0]		INT_CONFIG7 [1:0]		INT_CONFIG6 [1:0]		INT_CONFIG5 [1:0]		INT_CONFIG4 [1:0]		INT_CONFIG3 [1:0]		INT_CONFIG2 [1:0]		INT_CONFIG1 [1:0]		INT_CONFIG0 [1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0xC68-0xCFC	Reserved	Reserved																															
0xD00	GICD_PPISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPI3	PPI2	PPI1	PPI0	PPI4	PPI5	PPI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0									
0xD04 + 0x4*x (x=0 to 11) Last address: 0xD14	GICD_SPISRx	SPISRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD18-0xEFC	Reserved	Reserved																															
0xF00	GICD_SGIR	Res.	Res.	Res.	Res.	Res.	Res.	TARGETLISTFILTER [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPUTARGETLIST[1:0]		NSATT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SGIINTID[3:0]			
	Reset value							x	x							x	x	x												x	x	x	x
0xF04-0xF0C	Reserved	Reserved																															
0xF10 + 0x4*x (x=0 to 3) Last address: 0xF1C	GICD_ CPENDSGIRx	Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING3 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING2 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING0 [1:0]	
	Reset value							0	0							0	0							0	0							0	0
0xF20 + 0x4*x (x=0 to 3) Last address: 0xF2C	GICD_ SPENDSGIRx	Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING3 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING2 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING0 [1:0]	
	Reset value							0	0							0	0							0	0							0	0
0xF30-0xFCC	Reserved	Reserved																															
0xFD0	GICD_PIDR4	PIDR4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0xFD4 + 0x4*(x-5), (x=5 to 7) Last address: 0xFDC	GICD_PIDRx	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0	GICD_PIDR0	PIDR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
0xFE4	GICD_PIDR1	PIDR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0
0xFE8	GICD_PIDR2	PIDR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1
0xFEC	GICD_PIDR3	PIDR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 189. GICD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	GICD_CIDR0	CIDR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0xFF4	GICD_CIDR1	CIDR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0xFF8	GICD_CIDR2	CIDR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0xFFC	GICD_CIDR3	CIDR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

28.6 GIC CPU interface (GICC)

Each CPU interface provides a programming interface for the following actions:

- Enable the signaling of interrupt requests by the CPU interface.
- Acknowledge an interrupt.
- Indicate completion of the processing of an interrupt.
- Set an interrupt priority mask for the processor.
- Define the preemption policy for the processor.
- Determine the highest priority pending interrupt for the processor

All the registers are word-accessible.

Note: GICC offsets are provided with the default value of CA35SS_SYSCFG_GIC_CFGR (GICC set for a 64 KB granule system). Please refer to Arm documentation when set to 4 KB granule system.

28.6.1 GICC control register (GICC_CTLR)

Address offset: 0x000

Reset value: 0bXXXX XXXX XXXX XXXX XXXX X000 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EOIMODENS	EOIMODES	IRQBYPDISGRP1	FIQBYPDISGRP1	IRQBYPDISGRP0	FIQBYPDISGRP0	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **EOIMODENS**: Alias of EOIMODENS from the non-secure copy of this register.

Bit 9 **EOIMODES**:EOI mode for secure accesses

This bit controls the behavior of accesses to GICC_EOIR and GICC_DIR registers. This control applies only to secure accesses.

Bit 8 **IRQBYPDISGRP1**: Alias of IRQBYPDISGRP1 from the non-secure copy of this register.

Bit 7 **FIQBYPDISGRP1**: Alias of FIQBYPDISGRP1 from the non-secure copy of this register.

Bit 6 **IRQBYPDISGRP0**: IRQ bypass disable for group 0 interrupts

When the signaling of IRQs by the CPU interface is disabled, this bit partly controls whether the bypass IRQ signal is signaled to the processor.

Bit 5 **FIQBYPDISGRP0**: FIQ bypass disable for group 0 interrupts

When the signaling of FIQs by the CPU interface is disabled, this bit partly controls whether the bypass FIQ signal is signaled to the processor.

Bit 4 **CBPR**: BPR control

Controls whether GICC_BPR provides common control to group 0 and group 1 interrupts.

Bit 3 **FIQEN**: FIQ enable for group 0 interrupts

Controls whether the CPU interface signals group 0 interrupts to a target processor using the FIQ or the IRQ signal. The GIC always signals group 1 interrupts using the IRQ signal.

Bit 2 **ACKCTL**: Acknowledge control

When the highest priority pending interrupt is a group 1 interrupt, this bit determines both:
- whether a read of GICC_IAR acknowledges the interrupt, or returns a spurious interrupt ID
- whether a read of GICC_HPPIR returns the ID of the highest priority pending interrupt, or returns a spurious interrupt ID

Note: Arm deprecates use of this bit and strongly recommends using a software model where this bit is set to 0.

Bit 1 **ENABLEGRP1**: Enable group 1 interrupts

This bit enables the signaling of group 1 interrupts by the CPU interface to the connected processor.

Bit 0 **ENABLEGRP0**: Enable group 0 interrupts

This bit enables the signaling of group 0interrupts by the CPU interface to the connected processor.

28.6.2 GICC control non-secure access register (GICC_CTLRNS)

Address offset: 0x000

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XX0X X00X XXX0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	Res.	Res.	IRQBYPDISGRP1	FIQBYPDISGRP1	Res.	Res.	Res.	Res.	ENABLEGRP1
						rw			rw	rw					rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EOIMODENS**: EOI mode for non- secure accesses

This bit controls the behavior of accesses to GICC_EOIR and GICC_DIR registers.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **IRQBYPDISGRP1**: IRQ bypass for group 1 interrupts

When the signaling of IRQs by the CPU interface is disabled, this bit partly controls whether the bypass IRQ signal is signaled to the processor.

Bit 5 **FIQBYPDISGRP1**: FIQ bypass disable for group 1 interrupts

When the signaling of FIQs by the CPU interface is disabled, this bit partly controls whether the bypass FIQ signal is signaled to the processor.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **ENABLEGRP1**: Enable group1 interrupts

This bit enables the signaling of group 1 interrupts by the CPU interface to the connected processor.

28.6.3 GICC input priority mask register (GICC_PMR)

Address offset: 0x004

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX 0000 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: Priority mask level for the CPU interface

If the priority of an interrupt is higher than the value indicated by this bitfield, the interface signals the interrupt to the processor.

Bits 2:0 Reserved, must be kept at reset value.

28.6.4 GICC binary point register (GICC_BPR)

Address offset: 0x008

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX X010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

This bitfield value controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption and a subpriority field. Minimum value is 2.

28.6.5 GICC binary point non-secure access register (GICC_BPRNS)

Address offset: 0x008

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX X011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

This bitfield value controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption and a subpriority field. Minimum value is 3.

28.6.6 GICC interrupt acknowledge register (GICC_IAR)

Address offset: 0x00C

Reset value: 0bXXXX XXXX XXXX XXXX XXXX X011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**: CPU requesting the interrupt

For SGIs in a multiprocessor implementation, this bit identifies the processor that requested the interrupt. It returns the number of the CPU interface that made the request.

For example, a value of 1 means the request was generated by a write to the GICD_SGIR on CPU interface 1.

Bits 9:0 **INTERRUPT_ID[9:0]**: interrupt ID

28.6.7 GICC end of interrupt register (GICC_EOIR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the write refers to an SGI, this bit contains the CPUID value from the corresponding GICC_IAR access.

Bits 9:0 **EOIINTID[9:0]**: Interrupt ID value from the corresponding GICC_IAR access

28.6.8 GICC running priority register (GICC_RPR)

Address offset: 0x014

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								r	r	r	r	r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: Current running priority on the CPU interface

Bits 2:0 Reserved, must be kept at reset value.

28.6.9 GICC highest priority pending interrupt register (GICC_HPPIR)

Address offset: 0x018

Reset value: 0bXXXX XXXX XXXX XXXX XXXX 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the PENDINTID bitfield returns the ID of an SGI, this bit contains the CPUID value for this interrupt. This identifies the processor that generated the interrupt.

Bits 9:0 **PENDINTID[9:0]**: Interrupt ID of the highest-priority pending interrupt

28.6.10 GICC aliased binary point register (GICC_ABPR)

Address offset: 0x01C

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX X011

GICC_ABPR is an alias of the non-secure GICC_BPR. When CBPR is cleared to 0 in GICC_CTLR, a secure access to this register is equivalent to a non-secure access to GICC_BPR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

The value of this bitfield controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption, and a subpriority field. Minimum value is 3.

28.6.11 GICC aliased interrupt acknowledge register (GICC_AIAR)

Address offset: 0x020

Reset value: 0bXXXX XXXX XXXX XXXX XXXX X011 1111 1111

GICC_AIAR is an alias of the non-secure view of GICC_IAR. A secure access to this register is identical to a non-secure access to GICC_IAR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

For SGIs in a multiprocessor implementation, this bit identifies the processor that requested the interrupt. It returns the number of the CPU interface that made the request.

For example, a value of 1 means the request was generated by a write to the GICD_SGIR on CPU interface 1.

Bits 9:0 **INTERRUPT_ID[9:0]**: Interrupt ID

28.6.12 GICC aliased end of interrupt register (GICC_AEOIR)

Address offset: 0x024

Reset value: 0xXXXX XXXX

GICC_AEOIR is an alias of the non-secure GICC_EOIR. A secure access to this register is similar to a non-secure access to GICC_EOIR, except that EOIMODES in GICC_CTLR is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, when processing an SGI, this bit must contain the CPUID value from the corresponding GICC_AIAR, or non-secure GICC_IAR, access.

Bits 9:0 **EOIINTID[9:0]**:

This bitfield contains the interrupt ID value from the corresponding GICC_AIAR, or non-secure GICC_IAR, access.

28.6.13 GICC aliased highest priority pending interrupt register (GICC_AHPPIR)

Address offset: 0x028

Reset value: 0bXXXX XXXX XXXX XXXX XXX0 0011 1111 1111

This register is an alias of the non-secure GICC_HPPIR. A secure access to this register is equivalent to a non-secure access to GICC_HPPIR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the PENDINTID bitfield returns the ID of an SGI, this bit contains the CPUID value for that interrupt. This identifies the processor that generated the interrupt.

Bits 9:0 **PENDINTID[9:0]**: Interrupt ID of the highest-priority pending interrupt
This bitfield contains the interrupt ID of the highest-priority pending interrupt If that interrupt is a group 1 interrupt. Otherwise, its value is the spurious interrupt ID, 1023.

28.6.14 GICC active priority register (GICC_APR0)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]:** Active priority

28.6.15 GICC non-secure active priority register (GICC_NSAPR0)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSAPR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSAPR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NSAPR0[31:0]**: Non-secure active priority

28.6.16 GICC interface identification register (GICC_IIDR)

Address offset: 0x0FC

Reset value: 0x0202 143B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCTID[11:0]												ARCH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION[3:0]				IMPLEMENTER[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 **PRODUCTID[11:0]**: product ID

Bits 19:16 **ARCH[3:0]**: GIC architecture version

Bits 15:12 **REVISION[3:0]**: revision number for the CPU interface

Bits 11:0 **IMPLEMENTER[11:0]**: GIC implementer (0x43B Arm implementation)

28.6.17 GICC deactivate interrupt register (GICC_DIR)

Address offset: 0x10000

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

For an SGI in a multiprocessor implementation, this bit identifies the processor that requested the interrupt

Bits 9:0 **INTERRUPT_ID[9:0]**: interrupt ID

28.6.18 GICC register map

Table 190. GICC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICC_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	EOIMODES	IRQBYPDISGRP1	FIQBYPDISGRP1	IRQBYPDISGRP0	FIQBYPDISGRP0	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x000	GICC_CTLRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	Res.	Res.	IRQBYPDISGRP1	FIQBYPDISGRP1	Res.	Res.	Res.	Res.	ENABLEGRP1
	Reset value																							0			0	0					0
0x004	GICC_PMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x008	GICC_BPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	0
0x008	GICC_BPRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	1

Table 190. GICC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	GICC_IAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x010	GICC_EOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x014	GICC_RPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									1	1	1	1	1			
0x018	GICC_HPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x01C	GICC_ABPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT [2:0]		
	Reset value																														0	1	1
0x020	GICC_AIAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x024	GICC_AEOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x028	GICC_AHPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x02C-0x0CC	Reserved	Reserved																															
0x0D0	GICC_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4-0x0DC	Reserved	Reserved																															
0x0E0	GICC_NSAPR0	NSAPR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4-0x0F8	Reserved	Reserved																															
0x0FC	GICC_IIDR	PRODUCTID[11:0]												ARCH[3:0]				REVISION [3:0]				IMPLEMENTER[11:0]											
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x100-0xFFC	Reserved	Reserved																															
0x1000	GICC_DIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x

Refer to *Section 2.4* for the register boundary addresses.

28.7 GIC virtual interface control, common (GICH)

The GICH registers are management ones. Configuration software on the Cortex-A35 must ensure they are accessible only by a hypervisor or similar software.

All the registers are word-accessible.

28.7.1 GICH hypervisor control register (GICH_HCR)

Address offset: 0x000

Reset value: 0b0000 0XXX XXXX XXXX XXXX XXXX 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOICOUNT[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1 DIE	VGRP1 EIE	VGRP0 DIE	VGRP0 EIE	NPIE	LRENP IE	UIE	EN
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **EOICOUNT[4:0]**: End-of-interrupt counter

This bitfield counts the number of EOIs received that do not have a corresponding entry in the list registers. The virtual CPU interface increments this bitfield automatically when a matching EOI is received. When EOIs do not clear a bit in the active priorities register, GICH_APR does not cause an increment.

Although not possible under correct operation, if an EOI occurs when the value of this bitfield is 31, this bitfield wraps to 0.

The maintenance interrupt is asserted whenever this bitfield is non-zero, and LRENPIE = 1.

Bits 26:8 Reserved, must be kept at reset value.

Bit 7 **VGRP1DIE**: Virtual machine disable group 1 interrupt enable

This bit enables the signaling of a maintenance interrupt while signaling of group 1 interrupts from the virtual CPU interface to the connected virtual machine, is disabled.

Bit 6 **VGRP1EIE**: Virtual machine enable group 1 interrupt enable

This bit enables the signaling of a maintenance interrupt while signaling of group 1 interrupts from the virtual CPU interface to the connected virtual machine, is enabled.

Bit 5 **VGRP0DIE**: Virtual machine disable group 0 interrupt enable

This bit enables the signaling of a maintenance interrupt while signaling of group 0 interrupts from the virtual CPU interface to the connected virtual machine, is disabled.

Bit 4 **VGRP0EIE**: Virtual machine enable group 0interrupt enable

This bit enables the signaling of a maintenance interrupt while signaling of group 0 interrupts from the virtual CPU interface to the connected virtual machine, is enabled.

Bit 3 **NPIE**: No pending interrupt enable

This bit enables the signaling of a maintenance interrupt while no pending interrupts are present in the list registers.

Bit 2 **LRENPIE**: List register entry not present interrupt enable

This bit enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid list register entry for an EOI request.

Bit 1 **UIE**: Underflow interrupt enable.
This bit enables the signaling of a maintenance interrupt when the list registers are empty, or hold only one valid entry.

Bit 0 **EN**: Global enable bit for the virtual CPU interface

28.7.2 GICH VGIC type register (GICH_VTR)

Address offset: 0x004

Reset value: 0x9000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIBITS[2:0]			PREBITS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LISTREGS[4:0]				
											r	r	r	r	r

Bits 31:29 **PRIBITS[2:0]**: Priority bits
This bitfield indicates the number of priority bits implemented, minus one.
For example, 0x4 means 5 bits of priority, and 32 priority levels.

Bits 28:26 **PREBITS[2:0]**: Preemption bits
This bitfield indicates the number of preemption bits implemented, minus one.
For example, 0x4 means 5 bits of preemption, and 32 preemption levels

Bits 25:5 Reserved, must be kept at reset value.

Bits 4:0 **LISTREGS[4:0]**: List registers
This bitfield indicates the number of implemented list registers, minus one.
For example, 0x3 means 4 list registers.

28.7.3 GICH virtual machine control register (GICH_VMCR)

Address offset: 0x008

Reset value: 0b0000 0XXX 0100 11XX XXXX XX0X XXX0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VMPRIMASK[4:0]					Res.	Res.	Res.	VMBP[2:0]			VMABP[2:0]			Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	VEM	Res.	Res.	Res.	Res.	VMCBPR	VMFIQEN	VMACKCTL	VMGRP1EN	VMGRP0EN
						rw					rw	rw	rw	rw	rw

Bits 31:27 **VMPRIMASK[4:0]**: Alias of GICV_PMR.PRIORITY

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:21 **VMBP[2:0]**: Alias of GICV_BPR.BINARY_POINT

Bits 20:18 **VMABP[2:0]**: Alias of GICV_ABPR.BINARY_POINT.

Bits 17:10 Reserved, must be kept at reset value.

Bit 9 **VEM**: Alias of GICV_CTLR.EOIMODE

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **VMCBPR**: Alias of GICV_CTLR.CBPR

Bit 3 **VMFIQEN**: Alias of GICV_CTLR.FIQEN

Bit 2 **VMACKCTL**: Alias of GICV_CTLR.ACKCTL

Bit 1 **VMGRP1EN**: Alias of GICV_CTLR.ENABLEGRP1

Bit 0 **VMGRP0EN**: Alias of GICV_CTLR.ENABLEGRP0

28.7.4 GICH maintenance interrupt status register (GICH_MISR)

Address offset: 0x010

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1D	VGRP1E	VGRP0D	VGRP0E	NP	LRENP	U	EOI
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **VGRP1D**: Disabled group 1 maintenance interrupt
Asserted whenever GICH_HCR.VGRP1DIE is set and GICH_VMCR.VMGRP1EN = 0.

Bit 6 **VGRP1E**: Enabled group 1 maintenance interrupt
Asserted whenever GICH_HCR.VGRP1EIE is set and GICH_VMCR.VMGRP1EN = 1.

Bit 5 **VGRP0D**: Disabled group 0 maintenance interrupt
Asserted whenever GICH_HCR.VGRP0DIE is set and GICH_VMCR.VMGRP0EN = 0.

Bit 4 **VGRP0E**: Enabled group 0 maintenance interrupt
Asserted whenever GICH_HCR.VGRP0EIE is set and GICH_VMCR.VMGRP0EN = 1.

Bit 3 **NP**: No pending maintenance interrupt
Asserted whenever GICH_HCR.NPIE = 1 and no list register is in pending state.

Bit 2 **LRENP**: List register entry not present maintenance interrupt
Asserted whenever GICH_HCR.LRENPIE = 1 and GICH_HCR.EOICOUNT is non-zero.

Bit 1 **U**: Underflow maintenance interrupt
Asserted whenever GICH_HCR.UIE is set and if none, or only one, of the list register entries are marked as a valid interrupt (if the corresponding GICH_LRx.STATE bits do not equal 0x0).

Bit 0 **EOI**: End of interrupt maintenance interrupt
Asserted whenever at least one list register asserts an EOI Interrupt, which means at least one bit at 1 in GICH_EISRx.

28.7.5 GICH end of interrupt status register (GICH_EISR0)

Address offset: 0x020

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EISR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EISR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EISR0[31:0]**: End of interrupt status

28.7.6 GICH empty list status register (GICH_ELSR0)

Address offset: 0x030

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELSR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ELSR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ELSR0[31:0]**: Empty list status

28.7.7 GICH active priority register (GICH_APR0)

Address offset: 0x0F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]**: Active priority

28.7.8 GICH list register x (GICH_LRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HW	GRP1	STATE[1:0]		PRIORITY[4:0]					Res.	Res.	Res.	PHYSICALID[9:6]			
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYSICALID[5:0]						VIRTUALID[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HW**: Hardware interrupt

This bit indicates whether this virtual interrupt is an hardware interrupt (corresponding to a physical interrupt). Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the PHYSICALID bitfield indicates.

Bit 30 **GRP1**:

This bit indicates whether this virtual interrupt is a group 1 virtual interrupt.

Bits 29:28 **STATE[1:0]**: State of the interrupt

Bits 27:23 **PRIORITY[4:0]**: Priority of the interrupt

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:10 **PHYSICALID[9:0]**: Physical ID

This bitfield function depends on the value of the GICH_LR.HW bit, as follows.

1) When GICH_LR.HW = 0, bits [9:0] have the following meanings:

- [19] EOI when this interrupt triggers an EOI maintenance interrupt
- [12:10] CPUID If the interrupt has the virtual ID for an SGI, that is, 0-15. This bitfield shows the requesting CPU ID. This appears in the relevant field of the virtual machine interrupt acknowledge register, GICV_IAR or GICV_AIAR. Otherwise, this bitfield must be set to 0.

2) When GICH_LR.HW = 1, PHYSICALID[9:0] indicates the physical interrupt ID that the hypervisor forwards to the GICD.

If this bitfield value is 0-15, or 1020-1023, behavior is unpredictable.

If this bitfield value is 16-31, it applies to the PPI associated with the same physical CPUID as the virtual CPU interface requesting the deactivation.

Bits 9:0 **VIRTUALID[9:0]**: virtual ID

This ID is returned to the guest OS when the interrupt is acknowledged through GICV_IAR. Each valid interrupt stored in the list registers must have a unique virtual ID for that virtual CPU interface.

If this bitfield value is 1020-1023, behavior is unpredictable.

28.7.9 GICH register map

Table 191. GICH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICH_HCR	EOICOUNT[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1DIE	VGRP1EIE	VGRP0DIE	VGRP0EIE	NPIE	LRENPIE	UIE	EN
	Reset value	0	0	0	0	0																				0	0	0	0	0	0	0	0

Table 191. GICH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	GICH_VTR	PRIBITS [2:0]			PREBITS [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LISTREGS[4:0]				
	Reset value	1	0	0	1	0	0																						0	0	0	1	1
0x008	GICH_VMCR	VMPRIMASK [4:0]						Res.	Res.	Res.	VMBP [2:0]			VMABP [2:0]										VEM					VMCBPR	VMFIQEN	VMACKCTL	VMGRP1EN	VMGRP0EN
	Reset value	0	0	0	0	0				0	1	0	0	1	1									0					0	0	0	0	0
0x00C	Reserved	Reserved																															
0x010	GICH_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1D	VGRP1E	VGRP0D	VGRP0E	NP	LRENP	U	EOI
	Reset value																									0	0	0	0	0	0	0	0
0x014-0x01C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x020	GICH_EISR0	EISR0[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0
0x024-0x02C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x030	GICH_ELSR0	ELSR0[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1
0x034-0x0EC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0F0	GICH_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F4-0x0FC	Reserved	Reserved																															
0x100 + 0.x4*x (x=0 to 3) Last address 0x10C	GICH_LRx	HW	GRP1	STATE [1:0]			PRIORITY[4:0]				Res.	Res.	Res.	PHYSICALID[9:0]										VIRTUALID[9:0]										
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

28.8 GIC virtual CPU interface (GICV)

The GICV forwards virtual interrupts to a connected Cortex-A35 processor core, subject to the normal GIC handling and prioritization rules. The virtual interface control registers control GICV operation. The GICV uses the contents of the list registers to determine when to signal virtual interrupts. When a processor core accesses the GICV, the list registers are updated.

All the registers are word-accessible.

28.8.1 GICV virtual machine control register (GICV_CTLR)

Address offset: 0x000

Reset value: 0bXXXX XXXX XXXX XXXX XXXX X000 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EOIMODE	Res.	Res.	Res.	Res.	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
						rw					rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EOIMODE**: End of interrupt mode

This bit controls the behavior associated with GICV_EOIR, GICV_AEOIR, and GICV_DIR.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **CBPR**: BPR control

This bit controls whether GICV_BPR controls both group 0 and group 1 virtual interrupts.

Bit 3 **FIQEN**: FIQ enable for group 0 interrupts

This bit controls whether interrupts marked as group 0 are presented as virtual FIQs.

Bit 2 **ACKCTL**: Acknowledge control

This bit controls whether a read of GICV_IAR, when the highest priority pending interrupt is a group 1 interrupt, causes the CPU interface to acknowledge the interrupt.

Note: Arm deprecates use of this bit, and strongly recommends that software is written to operate with this bit always set to 0.

Bit 1 **ENABLEGRP1**:

This bit enables the signaling of group 1 virtual interrupts by the GICV to the virtual machine.

Bit 0 **ENABLEGRP0**:

This bit enables the signaling of group 0 virtual interrupts by the GICV to the virtual machine.

28.8.2 GICV VM priority mask register (GICV_PMR)

Address offset: 0x004

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX 0000 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: Priority mask level for the virtual CPU interface
Only virtual interrupts with higher priority than this bitfield value can be signaled to the processor core.

Bits 2:0 Reserved, must be kept at reset value.

28.8.3 GICV VM binary point register (GICV_BPR)

Address offset: 0x008

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX X010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:
This bitfield value controls how the 8-bit virtual interrupt priority field is split into a group priority field, used to determine virtual interrupt preemption, and a subpriority field. Minimum value is 2.

28.8.4 GICV VM interrupt acknowledge register (GICV_IAR)

Address offset: 0x00C

Reset value: 0bXXXX XXXX XXXX XXXX XXX0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:
If HW is cleared to 0 in GICH_LRx (indicating that the interrupt is triggered in software), the value of CPUID in GICH_LRx, that indicates the CPU ID, is returned in this bit.
Otherwise this bit is read as zero.

Bits 9:0 **INTERRUPT_ID[9:0]**: Interrupt ID

28.8.5 GICV VM end of interrupt register (GICV_EOIR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the write refers to an SGI, this bit contains the CPUID value from the corresponding GICV_IAR access.

Bits 9:0 **EOIINTID[9:0]**:

If HW is set to 1 in GICH_LRx (indicating a hardware interrupt), a deactivate request is sent to the physical distributor, identifying the physical ID from the corresponding bitfield in the list register.

28.8.6 GICV VM running priority register (GICV_RPR)

Address offset: 0x014

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								r	r	r	r	r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: Current running priority on the virtual CPU interface

Bits 2:0 Reserved, must be kept at reset value.

28.8.7 GICV VM highest priority pending interrupt register (GICV_HPPIR)

Address offset: 0x018

Reset value: 0bXXXX XXXX XXXX XXXX XXX0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if GICH_LRx.HW = 0, this bit contains the CPUID value for this virtual interrupt. This identifies the virtual processor that generated the virtual interrupt.

Bits 9:0 **PENDINTID[9:0]**:

The virtual interrupt ID of the highest-priority pending virtual interrupt

28.8.8 GICV VM aliased binary point register (GICV_ABPR)

Address offset: 0x01C

Reset value: 0bXXXX XXXX XXXX XXXX XXXX XXXX XXXX X011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

This bitfield value controls how the 8-bit virtual interrupt priority field is split into a group priority field, used to determine virtual interrupt preemption, and a subpriority field. Minimum value is 3.

28.8.9 GICV VM aliased interrupt register (GICV_AIAR)

Address offset: 0x020

Reset value: 0bXXXX XXXX XXXX XXXX XXX0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

If HW is cleared to 0 in GICH_LRx (indicating that the interrupt is triggered in software), the CPUID value in GICH_LRx, that indicates the CPU ID, is returned in this bit. Otherwise, this bit is read as zero.

Bits 9:0 **INTERRUPT_ID[9:0]**: interrupt ID

If GRP1 = 0 in GICH_LRx, the interrupt is group 0. The spurious interrupt ID 1023 is returned, and the interrupt is not acknowledged.

28.8.10 GICV VM aliased end of interrupt register (GICV_AEOIR)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the write refers to an SGI, this field contains the CPUID value from the corresponding GICV_IAR access.

Bits 9:0 **EOIINTID[9:0]**:

If HW is set to 1 in GICH_LRx (indicating an hardware interrupt), a deactivate request is sent to the physical distributor, identifying the physical ID from the corresponding field in the list register.

28.8.11 GICV VM aliased highest priority pending interrupt register (GICV_AHPPIR)

Address offset: 0x028

Reset value: 0bXXXX XXXX XXXX XXXX XXX0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

If HW is set to 1 in GICH_LRx (indicating an hardware interrupt), a deactivate request is sent to the physical distributor, identifying the physical ID from the corresponding field in the list register.

Bits 9:0 **PENDINTID[9:0]**:

This bitfield contains the virtual interrupt ID of the highest-priority pending virtual interrupt, if that virtual interrupt is a group 1 virtual interrupt.

Otherwise, it contains the spurious virtual interrupt ID (1023).

28.8.12 GICV VM active priority register (GICV_APR0)

Address offset: 0x0D0

Reset value: 0x0000 0000

This register is an alias of GICH_APR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]:** Active priority

28.8.13 GICV VM CPU interface identification register (GICV_IIDR)

Address offset: 0x0FC

Reset value: 0x0202 143B

This register is an alias of GICC_IIDR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IIDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IIDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IIDR[31:0]:** see GICC_IIDR description

28.8.14 GICV VM deactivate interrupt register (GICV_DIR)

Address offset: 0x1000

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

This bit identifies the processor core that requested the interrupt.

Bits 9:0 **INTERRUPT_ID[9:0]**: Interrupt ID

28.8.15 GICV register map

Table 192. GICV register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICV_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODE	Res.	Res.	Res.	Res.	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
	Reset value																							0					0	0	0	0	0
0x004	GICV_PMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									0	0	0	0	0			

Table 192. GICV register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	GICV_BPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	0
0x00C	GICV_IAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x010	GICV_EOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x014	GICV_RPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									1	1	1	1	1			
0x018	GICV_HPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x01C	GICV_ABPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	1
0x020	GICV_AIAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x024	GICV_AEOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x028	GICV_AHPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x02C-0x0CC	Reserved	Reserved																															
0x0D0	GICV_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4-0x0F8	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0FC	GICV_IIDR	IIDR[31:0]																															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x100-0xFFC	Reserved	Reserved																															
0x1000	GICV_DIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x

Refer to *Section 2.4* for the register boundary addresses.

29 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable and direct event inputs. It provides wake-up requests to the power control, and generates a combined secure/non-secure interrupt request to the CPUs NVIC and events to the CPUs event input. For each CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

EXTI1 and EXTI2 wake-up requests allow the system to be woken up from Stop modes, and the CPU to be woken up from CStop and CStandby modes.

The interrupt request and event request generation can also be used in Run modes.

EXTI1 and EXTI2 include the EXTI mux I/O port selection.

EXTI1: CPU1 = Cortex-A35 and CPU2 = Cortex-M33

EXTI2: CPU1 = Cortex-A35 and CPU2 = Cortex-M33

29.1 EXTI main features

The EXTI main features are the following:

- EXTI1 85, and EXTI2 64 input events supported
- EXTI1 and EXTI2 two CPUs supported
- Most event inputs allow to wake up the system.
- Some events can only be used to generate a CPU wake-up.
- Events which do not have an associated wake-up flag in the peripheral, have a flag in the EXTI, and generate a combined secure/non-secure interrupt to the CPUs from the EXTI.
- Events can be used to generate a CPU wake-up event.

The asynchronous event inputs are classified in two groups:

- configurable events (signals from I/Os or peripherals able to generate a pulse)

  These events have the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used to condition the CPU wake-up, interrupt and event generation
  - Individual interrupt lines per CPU
  - Software trigger possibility

- direct events (interrupt and wake-up sources from peripherals having an associated flag which requires to be cleared in the peripheral)

  These events have the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used to condition the CPU wake-up and event generation
  - No software trigger possibility
- Secure events
  - The access to control and configuration bits of secure input events can be made secure and or privilege.
- Resource isolation support
  - Support for CID resource identification per CPU, protecting any resource (register) from access by an other resource
- EXTI I/O port selection

29.2 EXTI implementation

Table 193. EXTI implementation

EXTI features(1)	EXTI1	EXTI2
EXTI I/O port selection	X	X
EXTI I/O port to interconnect other peripherals	X	X
Number of events	85	64
Number of CPUs	2	2
Security	X	X

1. X = supported.

29.3 EXTI block diagram

The EXTI consists of the following blocks (see *Figure 158*):

- register block accessed via an AHB interface, which contains all the EXTI registers
- event input trigger block which provides event input edge trigger logic
- masking block which provides the event input distribution to the different wake-up, interrupts, and event outputs, plus the masking of these
- EXTI mux which provides the I/O port selection on to the EXTI event signal

Figure 158. EXTI block diagram

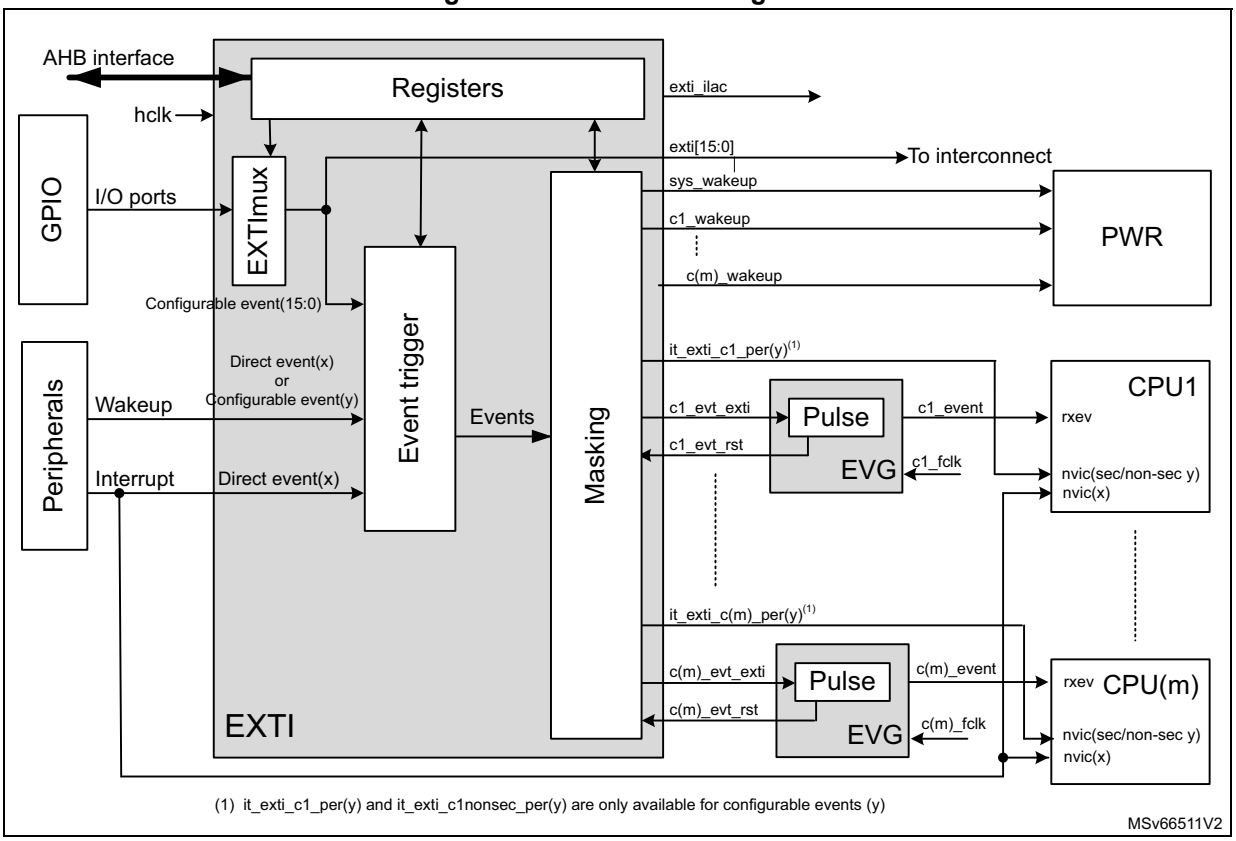


Table 194. EXTI signals

Name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to allow security, the AHB interface support secure accesses.
hclk	I	AHB bus clock and EXTI system clock.
exti_ilac	O	Illegal access event
Configurable event(y)	I	Asynchronous wake-up events from peripherals which do not have an associated interrupt and flag in the peripheral.
Direct event(x)	I	Synchronous and Asynchronous wake-up events from peripherals which have an associated interrupt and flag in the peripheral.
I/O ports	I	GPIO I/O ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals (only available for EXTI2)
it_exti_c(m)_per(y)	O	Combined secure/non-secure Interrupts to the CPU1 and CPU(m) associated with Configurable event (y)
c(m)_evt_exti	O	High-level sensitive event output for CPU(m) synchronous to hclk (EXTI1: m = 2, EXTI2: m = 2)
c(m)_evt_rst	I	Asynchronous reset input to clear c(m)_evt_exti (EXTI1: m = 2, EXTI2: m = 2)

Table 194. EXTI signals (continued)

Name	I/O	Description
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c(m)_wakeup	O	Wake-up request to PWR for CPU(m), synchronous to hclk (EXTI1: m = 1 to 2, EXTI2: m = 1 to 2)

Table 195. EVG signals

Name	I/O	Description
c_fclk	I	CPU free running clock.
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock.
c_event	O	Event pulse, synchronous to CPU clock.
c_evt_rst	O	Event reset signal, synchronous to CPU clock.

29.3.1 EXTI connections between peripherals and CPU

Peripherals able to generate wake-up or interrupt events when the system is in Stop mode or a CPU is in CStop mode, are connected to the EXTI.

Peripheral wake-up signals that generate a pulse, or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that interrupts the CPU.

Peripheral interrupt and wake-up signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

All GPIO ports are inputs for the EXTI multiplexer and used to select a port pin to wake up the system via a configurable event.

EXTI configurable event interrupts are connected to the respective NVIC of each CPU. A secure CPU receives both its secure and non-secure interrupts on the combined secure/non-secure it_exti_c(m)_per(y) interrupt signal. A non-secure CPU only receives its non-secure interrupts on the it_exti_c(m)_per(y) interrupt signal. Each CPU has its dedicated individual interrupt signal, which allows interrupt masking per CPU based on the EXTI_CmIMR settings.

The dedicated EXTI/EVG CPU event is connected to the respective CPU rxev input.

EXTI CPU wake-up signals are connected to the PWR, and are used to wake up the system and CPU subsystem bus clocks.

29.3.2 EXTI1 wake-up interrupt list

The wake-up sources are listed in the table below.

Some wake-up sources are able to generate an event to the CPU (see event column). The wake-up source capability to wake up a CPU is noted in wake-up column.

For CPU interrupt handling, see *Section 27: Nested vectored interrupt controller (NVIC)*.

Table 196. EXTI1 wake-up interrupts

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
0	EXTI0	GPIO EXTI line 0	Configurable	X	X	CPU1 and 2
1	EXTI1	GPIO EXTI line 1	Configurable	X	X	CPU1 and 2
2	EXTI2	GPIO EXTI line 2	Configurable	X	X	CPU1 and 2
3	EXTI3	GPIO EXTI line 3	Configurable	X	X	CPU1 and 2
4	EXTI4	GPIO EXTI line 4	Configurable	X	X	CPU1 and 2
5	EXTI5	GPIO EXTI line 5	Configurable	X	X	CPU1 and 2
6	EXTI6	GPIO EXTI line 6	Configurable	X	X	CPU1 and 2
7	EXTI7	GPIO EXTI line 7	Configurable	X	X	CPU1 and 2
8	EXTI8	GPIO EXTI line 8	Configurable	X	X	CPU1 and 2
9	EXTI9	GPIO EXTI line 9	Configurable	X	X	CPU1 and 2
10	EXTI10	GPIO EXTI line 10	Configurable	X	X	CPU1 and 2
11	EXTI11	GPIO EXTI line 11	Configurable	X	X	CPU1 and 2
12	EXTI12	GPIO EXTI line 12	Configurable	X	X	CPU1 and 2
13	EXTI13	GPIO EXTI line 13	Configurable	X	X	CPU1 and 2
14	EXTI14	GPIO EXTI line 14	Configurable	X	X	CPU1 and 2
15	EXTI15	GPIO EXTI line 15	Configurable	X	X	CPU1 and 2
16	PVD	PVD line	Configurable	-	X	CPU1 and 2
17	PVM	PVM line	Configurable	-	X	CPU1 and 2
18	RCC_MSI_FMON	RCC_MSI_FMON line	Direct	-	X	CPU1 and 2
19	RCC_HSI_FMON	RCC_HSI_FMON line	Direct	-	X	CPU1 and 2
20	Reserved	-	Direct	-	-	-
21	I2C1	I2C1 wake-up	Direct	-	X	CPU1 and 2
22	I2C2	I2C2 wake-up	Direct	-	X	CPU1 and 2
23	I2C3	I2C3 wake-up	Direct	-	X	CPU1 and 2
24	Reserved	-	Direct	-	-	-
25	Reserved	-	Direct	-	-	-
26	USART1	USART1 wake-up	Direct	-	X	CPU1 and 2

Table 196. EXTI1 wake-up interrupts (continued)

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
27	USART2	USART2 wake-up	Direct	-	X	CPU1 and 2
28	USART3	USART3 wake-up	Direct	-	X	CPU1 and 2
29	USART6	USART6 wake-up	Direct	-	X	CPU1 and 2
30	UART4	UART4 wake-up	Direct	-	X	CPU1 and 2
31	UART5	UART5 wake-up	Direct	-	X	CPU1 and 2
32	UART7	UART7 wake-up	Direct	-	X	CPU1 and 2
33	Reserved	-	Direct	-	-	-
34	Reserved	-	Direct	-	-	-
35	Reserved	-	Direct	-	-	-
36	SPI1	SPI1 wake-up	Direct	-	X	CPU1 and 2
37	SPI2	SPI2 wake-up	Direct	-	X	CPU1 and 2
38	SPI3	SPI3 wake-up	Direct	-	X	CPU1 and 2
39	SPI4	SPI4 wake-up	Direct	-	X	CPU1 and 2
40	SPI5	SPI5 wake-up	Direct	-	X	CPU1 and 2
41	SPI6	SPI6 wake-up	Direct	-	X	CPU1 and 2
42	Reserved	-	Direct	-	-	-
43	USBH	USBH wake-up	Configurable	-	X	CPU1 and 2
44	OTG_HS	OTG_HS wake-up	Configurable	-	X	CPU1 and 2
45	Reserved	-	Direct	-	-	-
46	Reserved	-	Direct	-	-	-
47	LPTIM1	LPtimer 1 wake-up	Direct	-	X	CPU1 and 2
48	LPTIM2	LPtimer 2 wake-up	Direct	-	X	CPU1 and 2
49	Reserved	-	Direct	-	-	-
50	Reserved	-	Direct	-	-	-
51	Reserved	-	Direct	-	-	-
52	WKUP1	PWR WKUP1 wake-up	Direct	-	X	CPU1 and 2
53	WKUP2	PWR WKUP2 wake-up	Direct	-	X	CPU1 and 2
54	WKUP3	PWR WKUP3 wake-up	Direct	-	X	CPU1 and 2
55	WKUP4	PWR WKUP4 wake-up	Direct	-	X	CPU1 and 2
56	WKUP5	PWR WKUP5 wake-up	Direct	-	X	CPU1 and 2
57	WKUP6	PWR WKUP6 wake-up	Direct	-	X	CPU1 and 2
58	Reserved	-	Direct	-	-	-

Table 196. EXTI1 wake-up interrupts (continued)

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
59	IPCC1 NS CPU1	IPCC1 CPU1 RX occupied and TX free non-secure interrupts	Direct	-	X	CPU1(2)
60	IPCC1 NS CPU2	IPCC1 CPU2 RX occupied and TX free non-secure interrupts	Direct	-	X	CPU2(3)
61	IPCC1 S CPU1	IPCC1 CPU1 RX occupied and TX free secure interrupts	Direct	-	X	CPU1(2)
62	IPCC1 S CPU2	IPCC1 CPU2 RX occupied and TX free secure interrupts	Direct	-	X	CPU2(3)
63	Reserved	-	Direct	-	-	-
64	C2SEV	CPU2 SEV event	Configurable	-	X	CPU1(2)
65	C1SEV	CPU1 SEV event	Configurable	X	X	CPU2(3)
66	Reserved	-	Direct	-	-	-
67	WWDG1 RST	WWDG1 reset	Configurable	-	X	CPU1(2)
68	ETH1_PMT wakeup	ETH1 PMT wake-up	Direct	-	X	CPU1 and 2
69	Reserved	-	Direct	-	-	-
70	ETH2_PMT wakeup	ETH2 PMT wake-up	Direct	-	X	CPU1 and 2
71	Reserved	-	Direct	-	-	-
72	DTS	DTS wake-up	Direct	-	X	CPU1 and 2
73	CPU2 SYSRESETREQ	CPU2 local reset event	Configurable	-	X	CPU1(2)
74	I3C1	I3C1 wake-up	Direct	-	X	CPU1 and 2
75	I3C2	I3C2 wake-up	Direct	-	X	CPU1 and 2
76	I3C3	I3C3 wake-up	Direct	-	X	CPU1 and 2
77	HPDMA1 CH 0 to 15 CPU1	HPDMA1 channel 0 to 15 CPU1 irq	Direct	-	X	CPU1(2)
78	HPDMA2 CH 0 to 15 CPU1	HPDMA2 channel 0 to 15 CPU1 irq	Direct	-	X	CPU1(2)
79	HPDMA3 CH 0 to 15 CPU1	HPDMA3 channel 0 to 15 CPU1 irq	Direct	-	X	CPU1(2)
80	HPDMA1 CH 0 to 15 CPU2	HPDMA1 channel 0 to 15 CPU2 irq	Direct	-	X	CPU2(3)
81	HPDMA2 CH 0 to 15 CPU2	HPDMA2 channel 0 to 15 CPU2 irq	Direct	-	X	CPU2(3)
82	HPDMA3 CH 0 to 15 CPU2	HPDMA3 channel 0 to 15 CPU2 irq	Direct	-	X	CPU2(3)
83	Reserved	-	Direct	-	-	-
84	Reserved	-	Direct	-	-	-

1. Events are only generated for CPU2.
2. For correct operation, the EXTI1 event EXTI1_C2IMRm.IMb bit must be set to 0 before CPU1 uses this event.
3. For correct operation, the EXTI1 event EXTI1_C1IMRm.IMb bit must be set to 0 before CPU2 uses this event.

29.3.3 EXTI2 wake-up interrupt list

The wake-up sources are listed in the table below.

Some wake-up sources are able to generate an event to the CPU (see event column). The wake-up source capability to wake up a CPU is noted in wake-up column.

For CPU interrupt handling, see *Section 27: Nested vectored interrupt controller (NVIC)*.

Table 197. EXTI2 wake-up interrupts

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
0	EXTI0	GPIO EXTI line 0	Configurable	X	X	CPU1 and 2
1	EXTI1	GPIO EXTI line 1	Configurable	X	X	CPU1 and 2
2	EXTI2	GPIO EXTI line 2	Configurable	X	X	CPU1 and 2
3	EXTI3	GPIO EXTI line 3	Configurable	X	X	CPU1 and 2
4	EXTI4	GPIO EXTI line 4	Configurable	X	X	CPU1 and 2
5	EXTI5	GPIO EXTI line 5	Configurable	X	X	CPU1 and 2
6	EXTI6	GPIO EXTI line 6	Configurable	X	X	CPU1 and 2
7	EXTI7	GPIO EXTI line 7	Configurable	X	X	CPU1 and 2
8	EXTI8	GPIO EXTI line 8	Configurable	X	X	CPU1 and 2
9	EXTI9	GPIO EXTI line 9	Configurable	X	X	CPU1 and 2
10	EXTI10	GPIO EXTI line 10	Configurable	X	X	CPU1 and 2
11	EXTI11	GPIO EXTI line 11	Configurable	X	X	CPU1 and 2
12	EXTI12	GPIO EXTI line 12	Configurable	X	X	CPU1 and 2
13	EXTI13	GPIO EXTI line 13	Configurable	X	X	CPU1 and 2
14	EXTI14	GPIO EXTI line 14	Configurable	X	X	CPU1 and 2
15	EXTI15	GPIO EXTI line 15	Configurable	X	X	CPU1 and 2
16	TAMP NS CPU1	TAMP CPU1 non-secure tamper wake-up	Direct	-	X	CPU1(2)
17	RTC NS CPU1	RTC CPU1 non-secure global wake-up	Direct	-	X	CPU1(2)
18	TAMP NS CPU2	TAMP CPU2 non-secure tamper wake-up	Direct	-	X	CPU2(3)
19	RTC NS CPU2	RTC CPU2 non-secure global wake-up	Direct	-	X	CPU2(3)
20	Reserved	-	Direct	-	-	-
21	TAMP S CPU1	TAMP CPU1 secure tamper wake-up	Direct	-	X	CPU1(2)
22	RTC S CPU1	RTC CPU1 secure global wake-up	Direct	-	X	CPU1(2)
23	TAMP S CPU2	TAMP CPU2 secure tamper wake-up	Direct	-	X	CPU2(3)
24	RTC S CPU2	RTC CPU2 secure global wake-up	Direct	-	X	CPU2(3)
25	Reserved	-	Direct	-	-	-

Table 197. EXTI2 wake-up interrupts (continued)

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
26	LPUART1	LPUART1 wake-up	Direct	-	X	CPU1 and 2
27	Reserved	-	Direct	-	-	-
28	Reserved	-	Direct	-	-	-
29	LPTIM3 wakeup	LPtimer 3 wake-up	Direct	-	X	CPU1 and 2
30	LPTIM4 wakeup	LPtimer 4 wake-up	Direct	-	X	CPU1 and 2
31	LPTIM5 wakeup	LPtimer 5 wake-up	Direct	-	X	CPU1 and 2
32	Reserved	-	Direct	-	-	-
33	Reserved	-	Direct	-	-	-
34	IWDG1 RST	IWDG1 reset	Configurable	-	X	CPU2(3)
35	IWDG2 RST	IWDG2 reset	Configurable	-	X	CPU1 and 2(4)
36	IWDG3 RST	IWDG3 reset	Configurable	-	X	CPU1(2)
37	IWDG4 RST	IWDG4 reset	Configurable	-	X	CPU1 and 2(4)
38	Reserved	-	Direct	-	-	-
39	IWDG1 EARLY WKUP	IWDG1 early wake-up	Direct	-	X	CPU1(2)
40	IWDG2 EARLY WKUP	IWDG2 early wake-up	Direct	-	X	CPU1(2)
41	IWDG3 EARLY WKUP	IWDG3 early wake-up	Direct	-	X	CPU2(3)
42	IWDG4 EARLY WKUP	IWDG4 early wake-up	Direct	-	X	CPU2(3)
43	Reserved	-	Direct	-	-	-
44	Reserved	-	Direct	-	-	-
45	Reserved	-	Direct	-	-	-
46	Reserved	-	Direct	-	-	-
47	Reserved	-	Direct	-	-	-
48	Reserved	-	Direct	-	-	-
49	IAC CPU1	IAC CPU1 interrupt	Direct	-	X	CPU1(2)
50	IAC CPU2	IAC CPU1 interrupt	Direct	-	X	CPU2(3)
51	VDDCPU_VD	PWR VDDCPU Voltage detection	Configurable	-	X	CPU1 and 2
52	VDDCORE_VD	PWR VDDCORE Voltage detection	Configurable	-	X	CPU1 and 2
53	RETRAM CRC ERR	RETRAM CRC error wake-up	Configurable	-	X	CPU1(2)
54	Reserved	-	Direct	-	-	-
55	Reserved	-	Direct	-	-	-
56	Reserved	-	Direct	-	-	-
57	Reserved	-	Direct	-	-	-
58	Reserved	-	Direct	-	-	-

Table 197. EXTI2 wake-up interrupts (continued)

EXTI n°	Acronym	Description	EXTI type	Event(1)	Security	Wake-up
59	Reserved	-	Direct	-	-	-
60	Reserved	-	Direct	-	-	-
61	Reserved	-	Direct	-	-	-
62	Reserved	-	Direct	-	-	-
63	CDBGPWRUPREQ	DEBUG power wake-up request	Direct	-	X	CPU1 and 2

1. Events are only generated for CPU2
2. For correct operation, the EXTI2 event EXTI2_C2IMRm.IM and EXTI2_C3IMRm.IMbit must be set to 0 before CPU1 uses this event.
3. For correct operation, the EXTI2 event EXTI2_C1IMRm.IM and EXTI2_C3IMRm.IM bit must be set to 0 before CPU2 uses this event.
4. For correct operation, the EXTI2 event, EXTI2_C3IMRm.IM and EXTI2_C3EMRm.EM bits must be set to 0 before CPU1 or CPU2 uses this event.

29.4 EXTI functional description

Depending on the EXTI event input type and wake-up targets, different logic implementations are used. The applicable features are controlled from register bits:

Table 198. EXTI event input configurations and register control

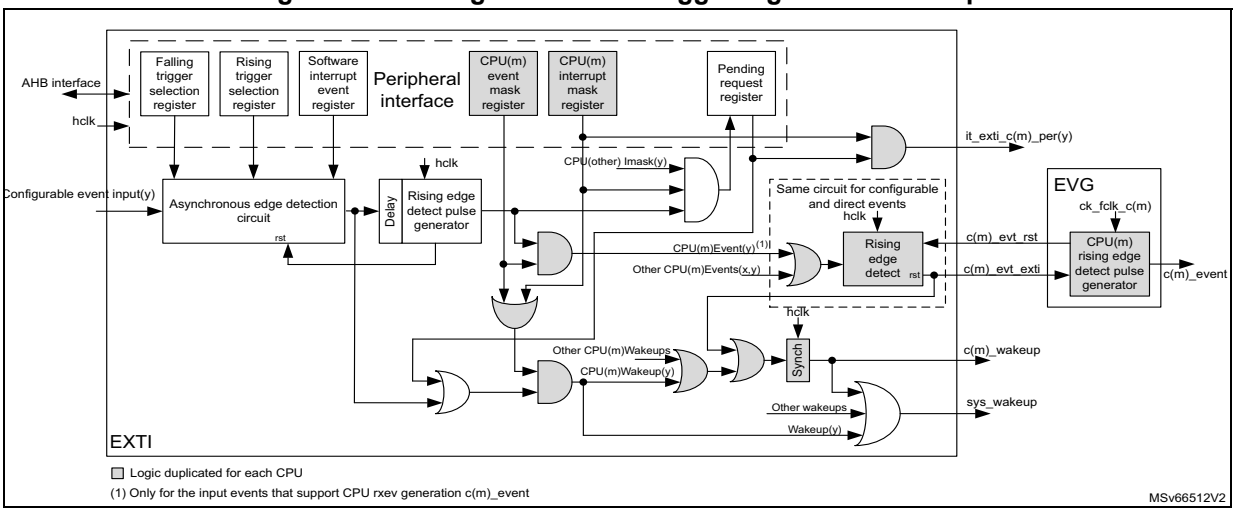
Event input type	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_RPR	EXTI_FPR	EXTI_SECCFGR	EXTI_PRIVCFGR	EXTI_CmIMR	EXTI_CmEMR(1)
Configurable	Configurable event input wake-up logic	x	x	x	x	x	x	x	x	x
Direct	Direct event input wake-up logic	-	-	-	-	-	x	x	x	x

1. Only for input events with configuration “rxev generation” enabled.

29.4.1 EXTI configurable event input wake-up

The figure below is a detailed representation of the logic associated with configurable event inputs which wake up the CPU(m) sub-system bus clocks and generate an EXTI pending flag and combined secure/non-secure interrupt to the CPU(m) and or a CPU(m) wake-up event.

Figure 159. Configurable event trigger logic CPU wake-up



The software interrupt event register is used to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers are used to enable and to select the configurable event active trigger edge or both edges.

Each CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask register. The enabled event is used to generate an event on the CPU. All events for a CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request register bits, shared by the CPUs. The pending register is only set for an unmasked interrupt. Each configurable event provides dedicated combined secure/non-secure interrupts to all CPUs. The configurable event interrupts need to be acknowledged by software in EXTI_RPR and/or EXTI_FPR.

When a CPU(m) wake-up (interrupt) or CPU(m) event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request can be cleared by any CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

29.4.2 EXTI direct event input wake-up

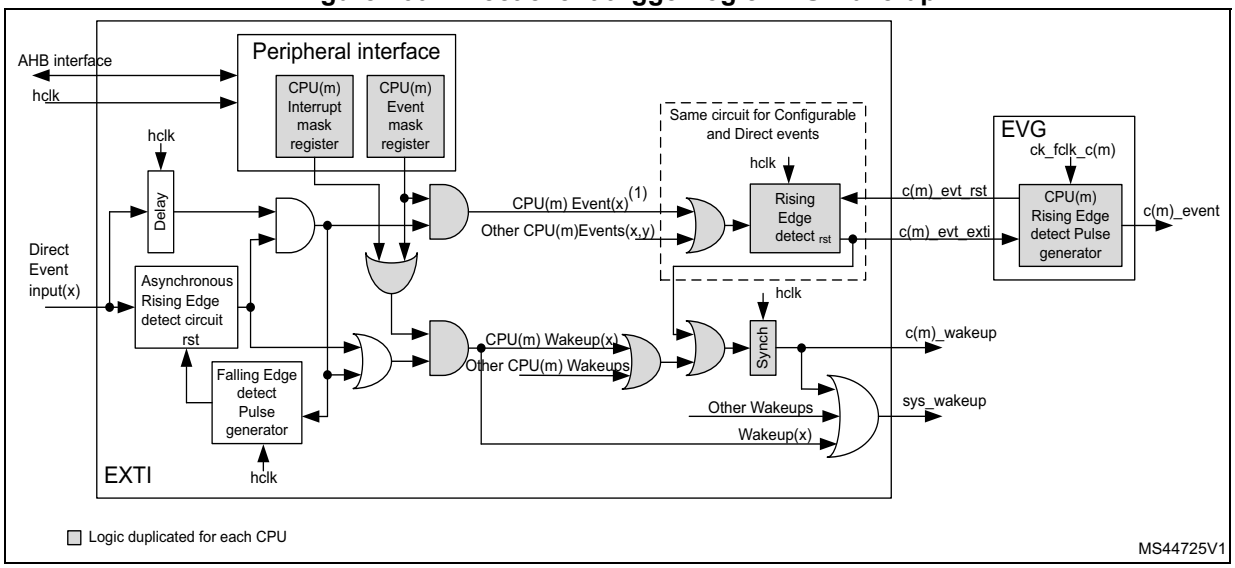
Figure 160 is a detailed representation of the logic associated with direct event inputs that wake up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks, and can generate a CPU wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to generate a CPU event that wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

Note: The direct events are cleared in the peripheral generating the event. When a direct event input enabled by CPU(m) is cleared by an other CPU before the CPU(m) clock is running, the CPU(m) no longer receives a CPU(m) interrupt nor CPU(m) event. The CPU(m) does not wake up. The system stays in Run mode, and generates the CPU(m) clock. For this reason, CPU(m) direct events must NOT be cleared by an other CPU.

Figure 160. Direct event trigger logic CPU wake-up

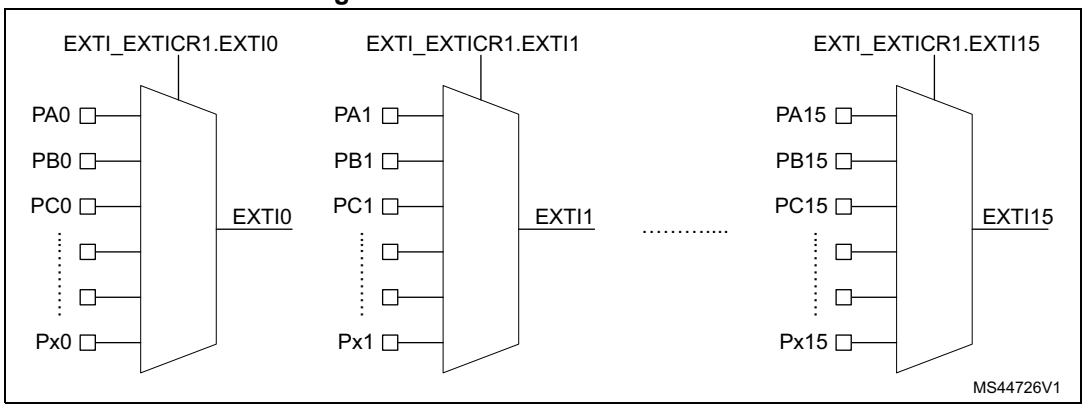


1. Only for the input events that support CPU rxev generation c(m)_event.

29.4.3 EXTI multiplexer selection

The EXTI multiplexer selection is available on EXTI1 and EXTI2. It is used to select GPIOs as interrupts and wake-up. The GPIOs are connected via 16 EXTI multiplexers to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI multiplexer output, is controlled by EXTIn_EXTICRx (x = 1 to 4).

Figure 161. EXTI mux GPIO selection



Only the EXTI mux outputs from EXTI2 are available as output signals to trigger other peripherals. The EXTI mux outputs are available independently from any masking in EXTI2_CmIMR1 and EXTI2_CmEMR1.

29.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wake-up event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU(m) wake-up is conditioned by the CPU(m) interrupt mask and CPU(m) event mask.

Table 199. Masking functionality

CPU interrupt enable EXTI_CmIMR.IMn	CPU event enable EXTI_CmEMR.EMn	Configurable event inputs EXTI_RPR.RPIFn EXTI_FPR.FPIFn	exti(n) interrupt(1)	CPU(m) event	CPU(m) wake-up
0 for all CPUs	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1 for any CPU	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The CPU dedicated exti(n), combined secure/non-secure interrupts, goes to each CPUs. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in EXTI_CmIMR.
2. Only if CPU(m) interrupt is enabled in EXTI_CmIMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU(m) interrupt is unmasked, the corresponding pending bit(s) EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set and

the CPU(m) subsystem is woken up, and the CPU combined secure/non-secure interrupt signal is activated. EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bit(s) must be cleared by software (writing them to 1). This clears the CPU interrupt. An interrupt is only generated, on the CPU dedicated interrupt line, when it is enabled in the CPU associated EXTI_CmIMR register.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU(m) interrupt is unmasked, the corresponding CPU subsystem is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU(m) event must be unmasked to generate an event. When the enabled edge(s) occur(s) on the event input, a CPU(m) event pulse is generated. There is no event pending bit.

For the configurable event inputs, an event request can be generated by software when writing 1 in EXTI_SWIER. This generates a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

29.6 EXTI event protection

The EXTI is able to protect event register bits from being modified by non-secure and unprivileged accesses. The protection can individually be activated per input event, via EXTI_SECCFGRx and EXTI_PRIVCFGRx. At EXTI level, the protection consists in preventing unauthorized write access to do the foollowing:

- Change settings of secure configurable events.
- Change masking of secure input events.
- Clear pending status of secure input events.

Note: The EXTI does not differentiate protected accesses from individual CPUs. When there is no CID resource isolation, then any CPU secure access can modify the protected register bits.

Table 200. Register protection overview

Register name	Access type	Security(1)(2)(3)(4)
EXTI_RTSRx	RW	Secure and privileged can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx, and CID filtering in EXTI_EnCIDCFGR.
EXTI_FTSRx	RW	
EXTI_SWIERx	RW	
EXTI_RPRx	RW	
EXTI_FPRx	RW	
EXTI_SECCFGRx	RW	Always secure and privileged, and CID filtering can bit wise be enabled in EXTI_EnCIDCFGR.
EXTI_PRIVCFGRx	RW	Always privileged, and secure can be enabled bit-wise in EXTI_SECCFGRx, and CID filtering in EXTI_EnCIDCFGR
EXTI_EXTICRx	RW	Secure and privileged can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx, and CID filtering in EXTI_EnCIDCFGR.
EXTI_LOCKR	RW	Always secure and privileged trusted domain CID

Table 200. Register protection overview (continued)

Register name	Access type	Security[1][2][3][4]
EXTI_CmIMRx	RW	Secure and privilege can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx, and CID filtering in EXTI_EnCIDCFGR. CPU(m) CID filtering is enabled on a register basis in EXTI_CmCIDCFGR.
EXTI_CmEMRx	RW	
EXTI_EnCIDCFGR	RW	Always secure and privileged trusted domain CID
EXTI_CmCIDCFGR	RW	Always secure and privileged trusted domain CID
EXTI_HWCFGRx	R	Non-secure unprivileged, no CID filtering.
EXTI_VER	R	
EXTI_ID	R	
EXTI_SID	R	

1. Security is enabled with EXTI_SECCFGRx registers.
2. Privilege is enabled with EXTI_PRIVCFGRx registers.
3. Resource compartment filtering is enabled with EXTI_EnCIDCFGR registers.
4. CPU(m) compartment filtering is enabled with EXTI_CmCIDCFGR registers.

29.6.1 EXTI register security protection

When security is enabled for an input event, the associated input event configuration and the control bits can only be modified and read by a secure access, a non-secure write access is discarded and a read returns 0.

When input events are non-secure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and non-secure access.

The security configuration in registers EXTI_SECCFGRx can be globally locked after reset by EXTI_LOCKR.GLOCK.

29.6.2 EXTI register privilege protection

When the privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privilege access, an unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privilege access and unprivileged access.

The privilege configuration in EXTI_PRIVCFGRx can be globally locked after reset by GLOCK in EXTI_LOCKR.

29.6.3 EXTI register CID protection

When the CID resource isolation is enabled for an input event, the associated input event configuration and control bits can only be modified and read by the assigned CID resource. Other CID resource write access is discarded, and a read returns 0.

When input events have no CID resource isolation, the CID filtering is disabled. Associated input event configuration and control bits can be modified and read by any access.

When the CPU CID resource isolation is enabled for a CPU, the associated CPU registers can only be modified and read by the assigned CID resource. Other CID resource write access is discarded, and a read returns 0.

When a CPU has no CID resource isolation, the CID filtering is disabled. The associated CPU registers can be modified and read by any access.

Table 201. Register protection hierarchy overview

TDCID filtered	CPU(m) CID filtered
EXTI_CmCIDCFGR = CPU(m) CID	EXTI_CmIMRn global register access
	EXTI_CmEMRn global register access
EXTI_EnCIDCFGR = CPU(m) CID	EXTI_RTSRn bit wise access filtering
	EXTI_FTSRn bit wise access filtering
	EXTI_SWIERn bit wise access filtering
	EXTI_RPRn bit wise access filtering
	EXTI_FPRn bit wise access filtering
	EXTI_SECCFGRn bit wise access filtering
	EXTI_PRIVCFGRn bit wise access filtering
	EXTI_CmIMRn bit wise access filtering
	EXTI_CmEMRn bit wise access filtering
	EXTI_EXTICRx bit filed wise access filtering

The security configuration in EXTI_EnCIDCFGRx and EXTI_CmCIDCFGR can be globally locked after reset by GLOCK in EXTI_LOCKR.

29.7 EXTIx registers

The EXTIx register map is divided in sections listed in the table below.

Table 202. EXTIx register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x040 - 0x05C	General configurable event [95:64] configuration
0x060 - 0x06C	EXTI I/O port mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU1 input event configuration
0x0C0 - 0x0FC	CPU2 input event configuration
0x180 - 0x2FC	Event input CID filtering configuration
0x300 - 0x30C	CPU CID filtering configuration

All registers can be accessed with word (32-bit), half-word (16-bit), and byte (8-bit) access.

For any DDCID access to registers, security and privileged policy applies. When DDEN = 0, accesses from the DDCID to CID filtered registers are ignored. When DDEN = 1, the DDCID has access to all registers regardless of any CID filtering policy.

29.7.1 EXTI1 rising trigger selection register (EXTI1_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT17	RT16
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 17 to 0)

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

When SECx is disabled in EXTI1_SECCFGR1, RTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, RTx can only be accessed with secure access. Non-secure write to RTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, RTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, RTx can only be accessed with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, RTx can only be accessed by the allowed processor CID. Illegal CID write to RTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

29.7.2 EXTI1 falling trigger selection register (EXTI1_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT17	FT16
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 17 to 0)

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line.

When SECx is disabled in EXTI1_SECCFGR1, FTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, FTx can only be accessed with secure access. Non-secure write to FTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, FTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, FTx can only be accessed with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, FTx can only be accessed by the allowed processor CID. Dis allowed CID CID write to FTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

29.7.3 EXTI1 software interrupt event register (EXTI1_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI17	SWI16
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **SWIx:** Software interrupt on event x (x = 17 to 0)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

When SECx is disabled in EXTI1_SECCFGR1, SWIx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, SWIx can only be accessed with secure access. Non-secure write to SWIx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1,SWIx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1,SWIx can only be accessed with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, SWIx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, SWIx can only be accessed by the allowed processor CID. Dis allowed CID write to SWIx is discarded and read returns 0.

A software interrupt is generated whatever the setting in EXTI1_RTSR and EXTI1_FTSR. It always returns 0 when read.

29.7.4 EXTI1 rising edge pending register (EXTI1_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF17	RPIF16
														rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **RPIFx:** Configurable event input x rising edge pending bit (x = 17 to 0)

This bit is set when the selected rising edge event or an EXTI1_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR1, RPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, RPIFx can only be accessed with secure access. Non-secure write to RPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, RPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, RPIFx can only be accessed with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, RPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to RPIFx is discarded, and read returns 0.

29.7.5 EXTI1 falling edge pending register (EXTI1_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF17	FPIF16
														rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **FPIFx:** configurable event input x falling edge pending bit (x = 17 to 0)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into this bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR1, FPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, FPIFx can only be accessed with secure access. Non-secure write to FPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, FPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, FPIFx can only be accessed with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, FPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to FPIFx is discarded, and read returns 0.

29.7.6 EXTI1 security configuration register (EXTI1_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security and privilege. A non-secure or unprivileged write access is ignored, and causes the generation of an secure illegal access event. All read return the register data.

Write to this register is ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	Res.	Res.	SEC23	SEC22	SEC21	Res.	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw			rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 **SECx:** Security enable on event input x (x = 31 to 26)
0: Event security disabled (non-secure)
1: Event security enabled (secure)
When CID filtering is disabled in EXTI1_ExCIDCFGR, SECx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, SECx can only be accessed by the allowed processor CID. Dis allowed CID write to SECx is discarded and read returns 0.

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:21 **SECx:** Security enable on event input x (x = 23 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **SECx:** Security enable on event input x (x = 19 to 0)

29.7.7 EXTI1 privilege configuration register (EXTI1_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged write access is discarded, and causes the generation of a privilege illegal access event. An unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	Res.	Res.	PRIV23	PRIV22	PRIV21	Res.	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw			rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 **PRIVx:** Privilege enable on event input x (x = 31 to 26)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)
When SECx is disabled in EXTI1_SECCFGR1, PRIVx can be accessed with secure and non-secure access.
When SECx is enabled in EXTI1_SECCFGR1, PRIVx can only be written with secure access. Non-secure write to PRIVx is discarded.
When CID filtering is disabled in EXTI1_ExCIDCFGR, PRIVx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, PRIVx can only be accessed by the allowed processor CID. Dis allowed CID write to PRIVx is discarded and read returns 0.

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:21 **PRIVx:** Privilege enable on event input x (x = 23 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **PRIVx:** Privilege enable on event input x (x = 19 to 0)

29.7.8 EXTI1 rising trigger selection register (EXTI1_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RT44	RT43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw											

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:11 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 44 to 43)

0: Rising trigger disabled (for event and Interrupt) for input line x

1: Rising trigger enabled (for event and Interrupt) for input line x

When SECx is disabled in EXTI1_SECCFGR2, RTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR2, RTx can only be accessed with secure access. Non-secure write to RTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR2, RTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR2, RTx can only be accessed with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, RTx can only be accessed by the allowed processor CID. Illegal CID write to RTx is discarded and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 10:0 Reserved, must be kept at reset value.

29.7.9 EXTI1 falling trigger selection register (EXTI1_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FT44	FT43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw											

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:11 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 44 to 43)

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

When SECx is disabled in EXTI1_SECCFGR2, FTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR2, FTx can only be accessed with secure access. Non-secure write to FTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR2, FTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR2, FTx can only be accessed with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, FTx can only be accessed by the allowed processor CID. Dis allowed CID write to FTx is discarded and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 10:0 Reserved, must be kept at reset value.

29.7.10 EXTI1 software interrupt event register (EXTI1_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SWI44	SWI43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw											

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:11 **SWIx:** Software interrupt on event x (x = 44 to 43)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

When SECx is disabled in EXTI1_SECCFGR2, SWIx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR2, SWIx can only be accessed with secure access. Non-secure write to SWIx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR2, SWIx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR2, SWIx can only be accessed with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, SWIx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, SWIx can only be accessed by the allowed processor CID. Dis allowed CID write to SWIx is discarded and read returns 0.

A software interrupt is generated whatever the setting in EXTI1_RTSR and EXTI1_FTSR. It always returns 0 when read.

Bits 10:0 Reserved, must be kept at reset value.

29.7.11 EXTI1 rising edge pending register (EXTI1_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RPIF44	RPIF43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rc_w1	rc_w1											

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:11 **RPIFx:** Configurable event input x rising edge pending bit (x = 44 to 43)

This bit is set when the selected rising edge event or an EXTI1_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR2, RPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR2, RPIFx can only be accessed with secure access. Non-secure write to RPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR2, RPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR2, this bit can only be accessed with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, this bit can only be accessed by the allowed processor CID. Dis allowed CID write to RPIFx is discarded and read returns 0.

Bits 10:0 Reserved, must be kept at reset value.

29.7.12 EXTI1 falling edge pending register (EXTI1_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FPIF44	FPIF43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rc_w1	rc_w1											

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:11 **FPIFx:** Configurable event input x falling edge pending bit (x = 44 to 43)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR2, FPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR2, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR2d, FPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR2, FPIFx can only be accessed with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, this bit can only be accessed by the allowed processor CID. Dis allowed CID write to FPIFx is discarded and read returns 0.

Bits 10:0 Reserved, must be kept at reset value.

29.7.13 EXTI1 security enable register (EXTI1_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security and privilege. A non-secure or unprivileged write access is ignored, and causes the generation of an secure illegal access event. All read return the register data.

Write to this register is ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SEC62	SEC61	SEC60	SEC59	Res.	SEC57	SEC56	SEC55	SEC54	SEC53	SEC52	Res.	Res.	Res.	SEC48
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC47	SEC46	Res.	SEC44	SEC43	Res.	SEC41	SEC40	SEC39	SEC38	SEC37	SEC36	Res.	Res.	Res.	SEC32
rw	rw		rw	rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **SECx:** Security enable on event input x (x = 62 to 59)
0: Event security disabled (non-secure)
1: Event security enabled (secure)
When CID filtering is disabled in EXTI1_ExCIDCFGR, SECx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, SECx can only be accessed by the allowed processor CID. Dis allowed CID write to SECx is discarded and read returns 0.

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **SECx:** Security enable on event input x (x = 57 to 52)

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **SECx:** Security enable on event input x (x = 48 to 46)

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **SECx:** Security enable on event input x (x = 44 to 43)

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SECx:** Security enable on event input x (x = 41 to 36)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEC32:** Security enable on event input 32

29.7.14 EXTI1 privilege enable register (EXTI1_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privilege write access protection. An unprivileged write access is discarded, and causes the generation of a privilege illegal access event. An unprivileged read returns the register data.

Write to this register is ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PRIV62	PRIV61	PRIV60	PRIV59	Res.	PRIV57	PRIV56	PRIV55	PRIV54	PRIV53	PRIV52	Res.	Res.	Res.	PRIV48
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV47	PRIV46	Res.	PRIV44	PRIV43	Res.	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	PRIV36	Res.	Res.	Res.	PRIV32
rw	rw		rw	rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **PRIVx:** Privilege enable on event input x (x = 62 to 59)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)
When SECx is disabled in EXTI1_SECCFGR2, PRIVx can be accessed with secure and non-secure access.
When SECx is enabled in EXTI1_SECCFGR2, PRIVx can only be accessed with secure access. Non-secure write to PRIVx is discarded.
When CID filtering is disabled in EXTI1_ExCIDCFGR, PRIVx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, PRIVx can only be accessed by the allowed processor CID. Dis allowed CID write to PRIVx is discarded, and read returns 0.

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **PRIVx:** Privilege enable on event input x (x = 57 to 52)

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **PRIVx:** Privilege enable on event input x (x = 48 to 46)

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PRIVx:** Privilege enable on event input x (x = 44 to 43)

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **PRIVx:** Privilege enable on event input x (x = 41 to 36)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PRIV32:** Privilege enable on event input 32

29.7.15 EXTI1 rising trigger selection register (EXTI1_RTSR3)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RT73	Res.	Res.	Res.	Res.	Res.	RT67	Res.	RT65	RT64
						rw						rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **RT73:** Rising trigger event configuration bit of configurable event input 73

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

When SECx is disabled in EXTI1_SECCFGR3, RTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR3, RTx can only be accessed with secure access. Non-secure write to RTx is discarded. Non-secure read returns 0.

When PRIVx i disabled in EXTI1_PRIVCFGR3, RTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR3, RTx can only be accessed with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, RTx can only be accessed by the allowed processor CID. Dis allowed CID write to RTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RT67:** Rising trigger event configuration bit of configurable event input 67

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 65 to 64)

29.7.16 EXTI1 falling trigger selection register (EXTI1_FTSR3)

Address offset: 0x044

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FT73	Res.	Res.	Res.	Res.	Res.	FT67	Res.	FT65	FT64
						rw						rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FT73:** Falling trigger event configuration bit of configurable event input 73

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line

When SECx is disabled in EXTI1_SECCFGR3, FTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR3, FTx can only be accessed with secure access. Non-secure write to FTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR3, FTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR3, FTx can only be accessed with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, FTx can only be accessed by the allowed processor CID. Dis allowed CID write to FTx is discarded and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **FT67:** Falling trigger event configuration bit of configurable event input 67

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 65 to 64)

29.7.17 EXTI1 software interrupt event register (EXTI1_SWIER3)

Address offset: 0x048

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SWI73	Res.	Res.	Res.	Res.	Res.	SWI67	Res.	SWI65	SWI64
						rw						rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **SWI73:** Software interrupt on event 73

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

When SECx is disabled in EXTI1_SECCFGR3, SWIx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR3, SWIx can only be accessed with secure access. Non-secure write to SWIx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR3, SWIx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR3, SWIx can only be accessed with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, SWIx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, SWIx can only be accessed by the allowed processor CID. Dis allowed CID write to SWIx is discarded and read returns 0.

A software interrupt is generated whatever the setting in EXTI1_RTSR and EXTI1_FTSR. It always returns 0 when read.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **SWI67:** Software interrupt on event 67.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SWIx:** Software interrupt on event x (x = 65 to 64)

29.7.18 EXTI1 rising edge pending register (EXTI1_RPR3)

Address offset: 0x04C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RPIF73	Res.	Res.	Res.	Res.	Res.	RPIF67	Res.	RPIF65	RPIF64
						rc_w1						rc_w1		rc_w1	rc_w1

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **RPIF73:** configurable event input 73 rising edge pending bit

This bit is set when the selected rising edge event or an EXTI1_SWIER Software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR3, RPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR3, RPIFx can only be accessed with secure access. Non-secure write to this RPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR3, RPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR3, RPIFx can only be accessed with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, RPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, RPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to RPIFx is discarded and read returns 0.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RPIF67:** configurable event input 67 rising edge pending bit

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RPIFx:** configurable event input x rising edge pending bit (x = 65 to 64)

29.7.19 EXTI1 falling edge pending register (EXTI1_FPR3)

Address offset: 0x050

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FPIF73	Res.	Res.	Res.	Res.	Res.	FPIF67	Res.	FPIF65	FPIF64
						rc_w1						rc_w1		rc_w1	rc_w1

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FPIF73:** configurable event input 73 falling edge pending bit

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

When SECx is disabled in EXTI1_SECCFGR3, FPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR3, FPIFx can only be accessed with secure access. Non-secure write to FPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR3, FPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR3, FPIFx can only be accessed with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, FPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, FPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to FPIFx is discarded and read returns 0.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **FPIF67:** configurable 67 falling edge pending bit

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FPIFx:** configurable event input x falling edge pending bit (x = 65 to 64)

29.7.20 EXTI1 security enable register (EXTI1_SECCFGR3)

Address offset: 0x054

Reset value: 0x0000 0000

This register provides write access security and privilege. A non-secure or unprivileged write access is ignored and causes the generation of an secure illegal access event. All reads return the register data.

Write to this register are ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC82	SEC81	SEC80
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC79	SEC78	SEC77	SEC76	SEC75	SEC74	SEC73	SEC72	Res.	SEC70	Res.	SEC68	SEC67	Res.	SEC65	SEC64
rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:8 **SECx:** Security enable on event input x (x = 82 to 72)
0: Event security disabled (non-secure)
1: Event security enabled (secure)
When CID filtering is disabled in EXTI1_ExCIDCFGR, SECx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, SECx can only be accessed by the allowed processor CID. Dis allowed CID write to SECx is discarded and read returns 0.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **SEC70:** Security enable on event input 70

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SECx:** Security enable on event input x (x = 68 to 67)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SECx:** Security enable on event input x (x = 65 to 64)

29.7.21 EXTI1 privilege enable register (EXTI1_PRIVCFGR3)

Address offset: 0x058

Reset value: 0x0000 0000

This register provides privilege write access protection. An unprivileged write access is discarded and cause the generation of an privilege illegal access event. A unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI1_LOCKR.

This register contains only bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV82	PRIV81	PRIV80
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV79	PRIV78	PRIV77	PRIV76	PRIV75	PRIV74	PRIV73	PRIV72	Res.	PRIV70	Res.	PRIV68	PRIV67	Res.	PRIV65	PRIV64
rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:8 **PRIVx:** Privilege enable on event input x (x = 82 to 72)

0: Event privilege disabled (unprivileged

1: Event privilege enabled (privileged)

When SECx is disabled in EXTI1_SECCFGR3, PRIVx can be accessed with secure and non-secure access.

When SECx is enabled in EXTI1_SECCFGR3, PRIVx can only be written with secure access. Non-secure write to PRIVx is discarded.

When CID filtering is disabled in EXTI1_ExCIDCFGR, PRIVx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, PRIVx can only be accessed by the allowed processor CID. Dis allowed CID write to PRIVx is discarded and read returns 0.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PRIV70:** Privilege enable on event input 70

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **PRIVx:** Privilege enable on event input x (x = 68 to 67)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PRIVx:** Privilege enable on event input x (x = 65 to 64)

29.7.22 EXTI1 external interrupt selection register 1 (EXTI1_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.
0x00: PA3 pin
0x01: PB3 pin
0x02: PC3 pin
0x03: PD3 pin
0x04: PE3 pin
0x05: PF3 pin
0x06: PG3 pin
0x07: PH3 pin
0x08: PI3 pin
0x09: PZ3 pin
Others: reserved

When SEC3 is disabled in EXTI1_SECCFGR1, this field can be accessed with non-secure and secure access.
When SEC3 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded. Non-secure read returns 0.
When PRIV3 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV3 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded, and read returns 0.
When CID filtering in EXTI1_E3CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E3CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
0x00: PA2 pin
0x01: PB2 pin
0x02: PC2 pin
0x03: PD2 pin
0x04: PE2 pin
0x05: PF2 pin
0x06: PG2 pin
0x07: PH2 pin
0x08: PI2 pin
0x09: PZ2 pin
Others: reserved

When SEC2 is disabled in EXTI1_SECCFGR1, this field can be accessed with non-secure and secure access.
When SEC2 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV2 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV2 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E2CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E2CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
0x00: PA1 pin
0x01: PB1 pin
0x02: PC1 pin
0x03: PD1 pin
0x04: PE1 pin
0x05: PF1 pin
0x06: PG1 pin
0x07: PH1 pin
0x08: PI1 pin
0x09: PZ1 pin
Others: reserved

When SEC1 is disabled in EXTI1_SECCFGR1, this field can be accessed with non-secure and secure access.
When SEC1 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV1 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV1 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E1CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E1CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection.

These bits are written by software to select the source input for EXTI0 external interrupt.
0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x04: PE0 pin
0x05: PF0 pin
0x06: PG0 pin
0x07: PH0 pin
0x08: PI0 pin
0x09: PZ0 pin
Others: reserved

When SEC0 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC0 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV0 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV0 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E0CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E0CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

29.7.23 EXTI1 external interrupt selection register 2 (EXTI1_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection.
These bits are written by software to select the source input for EXTI7 external interrupt.
0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
0x06: PG7 pin
0x07: PH7 pin
0x08: PI7 pin
0x09: PZ7 pin
Others: reserved

When SEC7 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC7 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV7 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV7 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E7CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E7CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection.
These bits are written by software to select the source input for EXTI6 external interrupt.
0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
0x05: PF6 pin
0x06: PG6 pin
0x07: PH6 pin
0x08: PI6 pin
0x09: PZ6 pin
Others: reserved

When SEC6 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC6 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV6 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV6 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E6CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E6CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection.

These bits are written by software to select the source input for EXTI5 external interrupt.
0x00: PA5 pin
0x01: PB5 pin
0x02: PC5 pin
0x03: PD5 pin
0x04: PE5 pin
0x05: PF5 pin
0x06: PG5 pin
0x07: PH5 pin
0x08: PI5 pin
0x09: PZ5 pin
Others: reserved

When SEC5 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC5 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV5 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV5 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E5CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E5CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection.

These bits are written by software to select the source input for EXTI4 external interrupt.
0x00: PA4 pin
0x01: PB4 pin
0x02: PC4 pin
0x03: PD4 pin
0x04: PE4 pin
0x05: PF4 pin
0x06: PG4 pin
0x07: PH4 pin
0x08: PI4 pin
0x09: PZ4 pin
Others: reserved

When SEC4 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC4 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV4 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV4 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E4CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E4CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

29.7.24 EXTI1 external interrupt selection register 3 (EXTI1_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection.

These bits are written by software to select the source input for EXTI11 external interrupt.
0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
0x07: PH11 pin
0x08: PI11 pin
0x09: PZ11 pin
Others: reserved

When SEC11 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC11 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV11 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV11 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E11CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E11CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection.

These bits are written by software to select the source input for EXTI10 external interrupt.
0x00: PA10 pin
0x01: PB10 pin
0x02: PC10 pin
0x03: PD10 pin
0x04: PE10 pin
0x05: PF10 pin
0x06: PG10 pin
0x07: PH10 pin
0x08: PI10 pin
0x09: PZ10 pin
Others: reserved

When SEC10 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC120 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV120 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV10 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E10CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E10CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection.

These bits are written by software to select the source input for EXTI9 external interrupt.
0x00: PA9 pin
0x01: PB9 pin
0x02: PC9 pin
0x03: PD9 pin
0x04: PE9 pin
0x05: PF9 pin
0x06: PG9 pin
0x07: PH9 pin
0x08: PI9 pin
0x09: PZ9 pin
Others: reserved

When SEC9 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC9 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV9 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV9 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E9CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E9CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection.

These bits are written by software to select the source input for EXTI8 external interrupt.
0x00: PA8 pin
0x01: PB8 pin
0x02: PC8 pin
0x03: PD8 pin
0x04: PE8 pin
0x05: PF8 pin
0x06: PG8 pin
0x07: PH8 pin
0x08: PI8 pin
0x09: PZ8 pin
Others: reserved

When SEC8 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC8 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV8 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV8 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E8CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E8CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

29.7.25 EXTI1 external interrupt selection register 4 (EXTI1_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection.

These bits are written by software to select the source input for EXTI15 external interrupt.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
0x06: PG15 pin
0x07: PH15 pin
0x08: PI15 pin
0x09: PZ15 pin
Others: reserved

When SEC15 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC15 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV15 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV15 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E15CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E15CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection.

These bits are written by software to select the source input for EXTI14 external interrupt.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
0x07: PH14 pin
0x08: PI14 pin
0x09: PZ14 pin
Others: reserved

When SEC14 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC14 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV14 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV14 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E14CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E14CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection.

These bits are written by software to select the source input for EXTI13 external interrupt.
0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
0x03: PD13 pin
0x04: PE13 pin
0x05: PF13 pin
0x06: PG13 pin
0x07: PH13 pin
0x08: PI13 pin
0x09: PZ13 pin
Others: reserved

When SEC13 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC13 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV13 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV13 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E13CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E13CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection.

These bits are written by software to select the source input for EXTI12 external interrupt.
0x00: PA12 pin
0x01: PB12 pin
0x02: PC12 pin
0x03: PD12 pin
0x04: PE12 pin
0x05: PF12 pin
0x06: PG12 pin
0x07: PH12 pin
0x08: PI12 pin
0x09: PZ12 pin
Others: reserved

When SEC12 is disabled in EXTI1_SECCFGR1 this field can be accessed with non-secure and secure access.
When SEC12 is enabled in EXTI1_SECCFGR1, this field can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.
When PRIV12 is disabled in EXTI1_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
When PRIV12 is enabled in EXTI1_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded and read returns 0.
When CID filtering in EXTI1_E12CIDCFGR is disabled, this field can be accessed by any processor CID.
When CID filtering in EXTI1_E12CIDCFGR is enabled, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded and read returns 0.

29.7.26 EXTI1 lock register (EXTI1_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register can only be written by the secure, privileged, trusted domain CID. A write access from any other protection level or domain CID is discarded and any read returns zero value. Both illegal write and read access generate an illegal access event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK:** Global security privilege and CID configuration registers EXTI1_SECCFGRx, EXTI1_PRIVCFGRx,EXTI1_EnCIDCFGR and EXTI1_CmCIDCFGR lock.

This bit is written once after reset.
0: Security, privilege, and CID configuration open can be modified.
1: Security, privilege, and CID configuration locked can no longer be modified.

29.7.27 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR1) (m = 1 to 2)

Address offset: Block 1: 0x080

Address offset: Block 2: 0x0C0

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

This register has CID resource protection, and can only be accessed by the allowed CID as defined in EXTI1_CmCIDCFGR. An illegal CID write access is discarded, and a read access return zero data, also on both illegal accesses an CID illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	Res.	Res.	IM23	IM22	IM21	Res.	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw			rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 31 to 26)

0: Wake-up and interrupt CPU(m) from input event x is masked.

1: Wake-up and interrupt CPU(m) from input event x is unmasked. CPU(m) wake-up event and configurable event interrupt generated.

When SECx is disabled in EXTI1_SECCFGR1, IMx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, IMx can only be accessed with secure access. Non-secure write to IMx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, IMx can be accessed with privileged and unprivileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, IMx can only be accessed with privileged access. Unprivileged write to IMx is discarded, and read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR IMx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR IMx can only be accessed by the allowed processor CID. Dis allowed CID write to IMx is discarded and read returns 0.

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:21 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 23 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 19 to 0)

29.7.28 EXTI1 CPU2 wake-up with event mask register (EXTI1_C2EMR1)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI1_C2CIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EMx:** CPU2 wake-up with event on event input x (x = 15 to 0)

0: Wake-up with event request from line x is masked.

1: Wake-up with event request from line x is unmasked.

When SECx is disabled in EXTI1_SECCFGR1, EMx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI1_SECCFGR1, EMx can only be accessed with secure access. Non-secure write to EMx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI1_PRIVCFGR1, EMx can be accessed with privileged and unprivileged access.

When PRIVx is enabled in EXTI1_PRIVCFGR1, EMx can only be accessed with privileged access. Unprivileged write to EMx is discarded, and read returns 0.

When CID filtering is disabled in EXTI1_ExCIDCFGR, EMx can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_ExCIDCFGR, EMx can only be accessed by the allowed processor CID. Disallowed CID write to EMx is discarded, and read returns 0.

29.7.29 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR2) (m = 1 to 2)

Address offset: Block 1: 0x090

Address offset: Block 2: 0x0D0

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI1_CmCIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IM62	IM61	IM60	IM59	Res.	IM57	IM56	IM55	IM54	IM53	IM52	Res.	Res.	Res.	IM48
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	Res.	IM44	IM43	Res.	IM41	IM40	IM39	IM38	IM37	IM36	Res.	Res.	Res.	IM32
rw	rw		rw	rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 62 to 59)
0: Wake-up and interrupt CPU(m) from input event x is masked.
1: Wake-up and interrupt CPU(m) from input event x is unmasked. CPU(m) wake-up event and configurable event interrupt generated.
When SECx is disabled in EXTI1_SECCFGR2, IMx can be accessed with non-secure and secure access.
When SECx is enabled in EXTI1_SECCFGR2, IMx can only be accessed with secure access. Non-secure write to IMx is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI1_PRIVCFGR2, IMx can be accessed with privileged and unprivileged access.
When PRIVx is enabled in EXTI1_PRIVCFGR2, IMx can only be accessed with privileged access. Unprivileged write to IMx is discarded, and read returns 0.
When CID filtering is disabled in EXTI1_ExCIDCFGR IMx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR IMx can only be accessed by the allowed processor CID. Dis allowed CID write to IMx is discarded, and read returns 0.

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 57 to 52)

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 48 to 46)

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 44 to 4**3)**

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 41 to 36)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **IM32:** CPUm wake-up with interrupt mask on event input 32

29.7.30 EXTI1 CPUm wake-up with interrupt mask register (EXTI1_CmIMR3) (m = 1 to 2)

Address offset: Block 1: 0x0A0

Address offset: Block 2: 0x0E0

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI1_CmCIDCFGR. An illegal CID write access is discarded and a read access returns zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM82	IM81	IM80
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM79	IM78	IM77	IM76	IM75	IM74	IM73	IM72	Res.	IM70	Res.	IM68	IM67	Res.	IM65	IM64
rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:8 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 82 to 72)
0: Wake-up and interrupt CPU(m) from input event x is masked
1: Wake-up and interrupt CPU(m) from input event x is unmasked. CPU(m) wake-up event and configurable event interrupt generated.
When SECx is disabled in EXTI1_SECCFGR3, IMx can be accessed with non-secure and secure access.
When SECx is enabled in EXTI1_SECCFGR3, IMx can only be accessed with secure access. Non-secure write to IMx is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI1_PRIVCFGR3, IMx can be accessed with privileged and unprivileged access.
When PRIVx is enabled EXTI1_PRIVCFGR3, IMx can only be accessed with privileged access. Unprivileged write to IMx is discarded and read returns 0.
When CID filtering is disabled in EXTI1_ExCIDCFGR, IMx can be accessed by any processor CID.
When CID filtering is enabled in EXTI1_ExCIDCFGR, IMx can only be accessed by the allowed processor CID. Dis allowed CID write to IMx is discarded, and read returns 0.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **IM70:** CPUm wake-up with interrupt mask on event input 70.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 68 to 67)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 65 to 64)

29.7.31 EXTI1 CPU2 wake-up with event mask register (EXTI1_C2EMR3)

Address offset: 0x0E4

Reset value: 0x0000 0000

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI1_C2CIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM65	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EM65:** CPU2 wake-up with event mask on event input 65

0: Wake-up with event request from line 65 is masked.

1: Wake-up with event request from line 65 is unmasked.

When SEC65 is disabled in EXTI1_SECCFGR3, EM65 can be accessed with non-secure and secure access.

When SEC65 is enabled in EXTI1_SECCFGR3, EM65 can only be accessed with secure access. Non-secure write to EM65 is discarded. Non-secure read returns 0.

When PRIV65 is disabled in EXTI1_PRIVCFGR3, EM65 can be accessed with privileged and unprivileged access.

When PRIV65 is enabled in EXTI1_PRIVCFGR3, EM65 can only be accessed with privileged access. Unprivileged write to EM65 is discarded and read returns 0.

When CID filtering is disabled in EXTI1_E65CIDCFGR, EM65 can be accessed by any processor CID.

When CID filtering is enabled in EXTI1_E65CIDCFGR, EM65 can only be accessed by the allowed processor CID. Disallowed CID write to EM65 is discarded and read returns 0.

Bit 0 Reserved, must be kept at reset value.

29.7.32 EXTI1 event CID configuration register (EXTI1_EnCIDCFGR)

Valid range: n = 0 to 19, 21 to 34, 36 to 50, 52 to 57, 59 to 62, 64 to 65, 67 to 68, 70, 72 to 84

Address offset: 0x180 + 0x04 * n (n = 0 to 84)

Reset value: 0x0000 0000

This register can only be written by the secure, privileged, trusted domain CID. A write access from any other protection level or domain CID is discarded and any read returns zero value. Both illegal write and read access generate an illegal access event.

Write to these registers are ignored if GLOCK is enabled in EXTI1_LOCKR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]:** EXTI event n allowed CID value

If CID filtering is disabled in CFEN, these bits are ignored, else EXTI event n allowed processor CID number.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** EXTI event n CID filtering enable

0: Disable EXTI event n filtering (EXTI event n can be accessed by any CIDs).

1: Enable EXTI event n filtering (EXTI event n can only be accessed by the processor with CID = CPUm CID).

29.7.33 EXTI1 processor m CID configuration register (EXTI1_CmCIDCFGR)

Address offset: 0x300 + 0x04 * (m - 1) (m = 1 to 2)

Reset value: 0x0000 0000

This register can only be written by the secure, privilege, trusted domain CID. An illegal write access to this register generates an illegal access event.

Write to these registers are ignored if GLOCK is enabled in EXTI1_LOCKR.

This registers have no read access limitations, and can be read by any domain CID, and any access protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]:** CPUm CID domain identification
If CFEN = 0, these bits are ignored, and there is no CPUm CID filtering, else CPUm CID domain identification number.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** CID filtering enabled for CPUm EXTI1_CmIMRx and EXTI1_CmEMRx registers
0: Disable the CID filtering for CPUm.
1: Enable the CID filtering for CPUm.

29.7.34 EXTI1 hardware configuration registers 13 (EXTI1_HWCFGR13)

Address offset: 0x3C0

Reset value: 0x0007 FF5B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[95:64]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged, and no resource filtering event
1: Security, privileged, and resource filtering capable event

29.7.35 EXTI1 hardware configuration registers 12 (EXTI1_HWCFGR12)

Address offset: 0x3C4

Reset value: 0x7BF1 DBF1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[63:32]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged, and no resource filtering event
1: Security, privileged, and resource filtering capable event

29.7.36 EXTI1 hardware configuration registers 11 (EXTI1_HWCFGR11)

Address offset: 0x3C8

Reset value: 0xFCEF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[31:0]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged, and no resource filtering event
1: Security, privileged, and resource filtering capable event

29.7.37 EXTI1 hardware configuration registers 7 (EXTI1_HWCFGR7)

Address offset: 0x3D8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[95:64]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.38 EXTI1 hardware configuration registers 6 (EXTI1_HWCFGR6)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[63:32]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.39 EXTI1 hardware configuration registers 5 (EXTI1_HWCFGR5)

Address offset: 0x3E0

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[31:0]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.40 EXTI1 hardware configuration registers 4 (EXTI1_HWCFGR4)

Address offset: 0x3E4

Reset value: 0x0000 020B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[95:64]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.41 EXTI1 hardware configuration registers 3 (EXTI1_HWCFGR3)

Address offset: 0x3E8

Reset value: 0x0000 1800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[63:32]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.42 EXTI1 hardware configuration registers 2 (EXTI1_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0003 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[31:0]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.43 EXTI1 hardware configuration register 1 (EXTI1_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0309 2154

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CIDWIDTH[3:0]				NBIOPORT[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVTEN[3:0]				NBCPUS[3:0]				NBEVENTS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CIDWIDTH[3:0]:** CID parameters bit width

Bits 23:16 **NBIOPORT[7:0]:** Hardware configuration of number of I/O ports on EXTI (n+1)

Bits 15:12 **CPUEVTEN[3:0]:** Hardware configuration of CPU(m) event output enable.

Bits 11:8 **NBCPUS[3:0]:** Hardware configuration number of CPUs (n+1)

Bits 7:0 **NBEVENTS[7:0]:** Hardware configuration number of event (n+1)

29.7.44 EXTI1 version register (EXTI1_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision number

Bits 3:0 **MINREV[3:0]:** Minor revision number

29.7.45 EXTI1 identification register (EXTI1_IPIDR)

Address offset: 0x3F8

Reset value: 0x000E 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]:** EXTI1 identification

29.7.46 EXTI1 size identification register (EXTI1_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

29.7.47 EXTI2 rising trigger selection register (EXTI2_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 15 to 0)

0: Rising trigger disabled (for event and interrupt) for input line x

1: Rising trigger enabled (for event and interrupt) for input line x

When SECx is disabled in EXTI2_SECCFGR1, RTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, RTx can only be accessed with secure access. Non-secure write to RTx x is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, RTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, RTx can only be accessed with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, RTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, RTx can only be accessed by the allowed processor CID. Illegal CID write to RTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

29.7.48 EXTI2 falling trigger selection register (EXTI2_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 15 to 0)

0: Falling trigger disabled (for event and Interrupt) for input line x

1: Falling trigger enabled (for event and Interrupt) for input line x

When SECx is disabled in EXTI2_SECCFGR1, FTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, FTx can only be accessed with secure access. Non-secure write to FTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, FTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, FTx can only be accessed with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, FTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, FTx can only be accessed by the allowed processor CID. Dis allowed CID write to FTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

29.7.49 EXTI2 software interrupt event register (EXTI2_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SWIx:** Software interrupt on event x (x = 15 to 0)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

When SECx is disabled in EXTI2_SECCFGR1, SWIx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, SWIx can only be accessed with secure access. Non-secure write to SWIx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, SWIx can be accessed with unprivileged and privileged access.

When PRIVx is disabled in EXTI2_PRIVCFGR1, SWIx can only be accessed with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, SWIx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, SWIx can only be accessed by the allowed processor CID. Dis allowed CID write to SWIx is discarded, and read returns 0.

A software interrupt is generated independent from the setting in EXTI2_RTSR and EXTI2_FTSR. It always returns 0 when read.

29.7.50 EXTI2 rising edge pending register (EXTI2_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RPIFx:** Configurable event input x rising edge pending bit (x = 15 to 0)

This bit is set when the selected rising edge event, or an EXTI2_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

When SECx is disabled in EXTI2_SECCFGR1, RPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, RPIFx can only be accessed with secure access. Non-secure write to RPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, RPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, RPIFx can only be accessed with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, RPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, RPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to RPIFx is discarded, and read returns 0.

29.7.51 EXTI2 falling edge pending register (EXTI2_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **FPIFx:** Configurable event input x falling edge pending bit (x = 15 to 0)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

When SECx is disabled in EXTI2_SECCFGR1, FPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, FPIFx can only be accessed with secure access. Non-secure write to FPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, FPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, FPIFx can only be accessed with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, FPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, FPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to FPIFx is discarded and read returns 0.

29.7.52 EXTI2 security configuration register (EXTI2_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security and privilege.A non-secure or unprivileged write access is ignored and causes the generation of an illegal access event. All read return the register data.

Write to this register are ignored if GLOCK is enabled in EXTI2_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	Res.	Res.	SEC26	Res.	SEC24	SEC23	SEC22	SEC21	Res.	SEC19	SEC18	SEC17	SEC16
rw	rw	rw			rw		rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **SECx:** Security enable on event input x (x = 31 to 29)
0: Event security disabled (non-secure)
1: Event security enabled (secure)
When CID filtering is disabled in EXTI2_ExCIDCFGR, SECx can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, SECx can only be accessed by the allowed processor CID. Dis allowed CID write to SECx is discarded and read returns 0.

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **SEC26:** Security enable on event input 26

Bit 25 Reserved, must be kept at reset value.

Bits 24:21 **SECx:** Security enable on event input x (x = 24 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **SECx:** Security enable on event input x (x = 19 to 0)

29.7.53 EXTI2 privilege configuration register (EXTI2_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privilege write access protection. An unprivileged write access is discarded and causes the generation of an illegal access event. A unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI2_LOCKR.

This register contains only bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	Res.	Res.	PRIV26	Res.	PRIV24	PRIV23	PRIV22	PRIV21	Res.	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw			rw		rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **PRIVx:** Privilege enable on event input x (x = 31 to 29)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)
When SECx is disabled in EXTI2_SECCFGR1, PRIVx can be accessed with secure and non-secure access.
When SECx is enabled in EXTI2_SECCFGR1, PRIVx can only be written with secure access. Non-secure write to PRIVx is discarded.
When CID filtering is disabled in EXTI2_ExCIDCFGR, PRIVx can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, PRIVx can only be accessed by the allowed processor CID. Dis allowed CID write to PRIVx is discarded, and read returns 0.

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **PRIV26:** Privilege enable on event input 26

Bit 25 Reserved, must be kept at reset value.

Bits 24:21 **PRIVx:** Privilege enable on event input x (x = 24 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **PRIVx:** Privilege enable on event input x (x = 19 to 0)

29.7.54 EXTI2 rising trigger selection register (EXTI2_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT53	RT52	RT51	Res.	Res.	Res.
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT37	RT36	RT35	RT34	Res.	Res.
										rw	rw	rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 53 to 51)

0: Rising trigger disabled (for event and interrupt) for input line x

1: Rising trigger enabled (for event and interrupt) for input line x

When SECx is disabled in EXTI2_SECCFGR2, RTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, RTx can only be accessed with secure access. Non-secure write to RTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, RTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, RTx can only be accessed with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, RTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, RTx can only be accessed by the allowed processor CID. Dis allowed CID write to RTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:2 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.55 EXTI2 falling trigger selection register (EXTI2_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT53	FT52	FT51	Res.	Res.	Res.
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT37	FT36	FT35	FT34	Res.	Res.
										rw	rw	rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 53 to 51)

0: Falling trigger disabled (for event and interrupt) for input line x

1: Falling trigger enabled (for event and interrupt) for input line x

When SECx is disabled in EXTI2_SECCFGR2, FTx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, FTx can only be accessed with secure access. Non-secure write to FTx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, FTx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, FTx can only be accessed with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, FTx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, FTx can only be accessed by the allowed processor CID. Dis allowed CID write to FTx is discarded, and read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:2 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.56 EXTI2 software interrupt event register (EXTI2_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI53	SWI52	SWI51	Res.	Res.	Res.
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI37	SWI36	SWI35	SWI34	Res.	Res.
										rw	rw	rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **SWIx:** Software interrupt on event x (x = 53 to 51)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

When SECx is disabled in EXTI2_SECCFGR2, SWIx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, SWIx can only be accessed with secure access. Non-secure write to SWIx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, SWIx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, SWIx can only be accessed with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, SWIx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, SWIx can only be accessed by the allowed processor CID. Dis allowed CID write to SWIx is discarded, and read returns 0.

A software interrupt is generated independent from the setting in EXTI2_RTSR and EXTI2_FTSR. It always returns 0 when read.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:2 **SWIx:** Software interrupt on event x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.57 EXTI2 rising edge pending register (EXTI2_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF53	RPIF52	RPIF51	Res.	Res.	Res.
										rc_w1	rc_w1	rc_w1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF37	RPIF36	RPIF35	RPIF34	Res.	Res.
										rc_w1	rc_w1	rc_w1	rc_w1		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **RPIFx:** Configurable event input x rising edge pending bit (x = 53 to 51)

This bit is set when the selected rising edge event or an EXTI2_SWIER Software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

When SECx is disabled in EXTI2_SECCFGR2, RPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, RPIFx can only be accessed with secure access. Non-secure write to RPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, RPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, RPIFx can only be accessed with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, RPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, RPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to RPIFx is discarded, and read returns 0.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:2 **RPIFx:** Configurable event input x rising edge pending bit (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.58 EXTI2 falling edge pending register (EXTI2_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF53	FPIF52	FPIF51	Res.	Res.	Res.
										rc_w1	rc_w1	rc_w1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF37	FPIF36	FPIF35	FPIF34	Res.	Res.
										rc_w1	rc_w1	rc_w1	rc_w1		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **FPIFx:** Configurable event input x pending bit (x = 53 to 51)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

When SECx is disabled in EXTI2_SECCFGR2, FPIFx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, FPIFx can only be accessed with secure access. Non-secure write to FPIFx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, FPIFx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, FPIFx can only be accessed with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, FPIFx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, FPIFx can only be accessed by the allowed processor CID. Dis allowed CID write to FPIFx is discarded, and read returns 0.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:2 **FPIFx:** Configurable event input x pending bit (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.59 EXTI2 security enable register (EXTI2_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security and privilege. A non-secure or unprivileged write access is ignored and causes the generation of an illegal access event. All read return the register data.

Write to this register are ignored if GLOCK is enabled in EXTI2_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC 63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC 53	SEC 52	SEC 51	SEC 50	SEC 49	Res.
rw										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SEC 42	SEC 41	SEC 40	SEC 39	Res.	SEC 37	SEC 36	SEC 35	SEC 34	Res.	Res.
					rw	rw	rw	rw		rw	rw	rw	rw		

Bit 31 **SEC63:** Security enable on event input 63

0: Event security disabled (non-secure)

1: Event security enabled (secure)

When CID filtering is disabled in EXTI2_ExCIDCFGR, SECx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, SECx can only be accessed by the allowed processor CID. Dis allowed CID write to SECx is discarded, and read returns 0.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **SECx:** Security enable on event input x (x = 53 to 49)

Bits 16:11 Reserved, must be kept at reset value.

Bits 10:7 **SECx:** Security enable on event input x (x = 42 to 39)

Bit 6 Reserved, must be kept at reset value.

Bits 5:2 **SECx:** Security enable on event input x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.60 EXTI2 privilege enable register (EXTI2_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privilege write access protection. An unprivileged write access is discarded and causes the generation of an illegal access event. A unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI2_LOCKR.

This register contains only bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV53	PRIV52	PRIV51	PRIV50	PRIV49	Res.
rw										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	PRIV39	Res.	PRIV37	PRIV36	PRIV35	PRIV34	Res.	Res.
					rw	rw	rw	rw		rw	rw	rw	rw		

Bit 31 **PRIV63:** Privilege enable on event input 63

0: Event privilege disabled (unprivileged)

1: Event privilege enabled (privileged)

When SECx is disabled in EXTI2_SECCFGR2, PRIVx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, PRIVx can only be accessed with secure access. Non-secure write to PRIVx is discarded. Non-secure read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, PRIVx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, PRIVx can only be accessed by the allowed processor CID. Dis allowed CID write to PRIVx is discarded, and read returns 0.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **PRIVx:** Privilege enable on event input x (x = 53 to 49)

Bits 16:11 Reserved, must be kept at reset value.

Bits 10:7 **PRIVx:** Privilege enable on event input x (x = 42 to 39)

Bit 6 Reserved, must be kept at reset value.

Bits 5:2 **PRIVx:** Privilege enable on event input x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.61 EXTI2 external interrupt selection register 1 (EXTI2_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.
0x00: PA3 pin
0x01: PB3 pin
0x02: PC3 pin
0x03: PD3 pin
0x04: PE3 pin
0x05: PF3 pin
0x06: PG3 pin
0x07: PH3 pin
0x08: PI3 pin
0x09: PZ3 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
0x00: PA2 pin
0x01: PB2 pin
0x02: PC2 pin
0x03: PD2 pin
0x04: PE2 pin
0x05: PF2 pin
0x06: PG2 pin
0x07: PH2 pin
0x08: PI2 pin
0x09: PZ2 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
0x00: PA1 pin
0x01: PB1 pin
0x02: PC1 pin
0x03: PD1 pin
0x04: PE1 pin
0x05: PF1 pin
0x06: PG1 pin
0x07: PH1 pin
0x08: PI1 pin
0x09: PZ1 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x04: PE0 pin
0x05: PF0 pin
0x06: PG0 pin
0x07: PH0 pin
0x08: PI0 pin
0x09: PZ0 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

29.7.62 EXTI2 external interrupt selection register 2 (EXTI2_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
0x06: PG7 pin
0x07: PH7 pin
0x08: PI7 pin
0x09: PZ7 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
0x05: PF6 pin
0x06: PG6 pin
0x07: PH6 pin
0x08: PI6 pin
0x09: PZ6 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
0x00: PA5 pin
0x01: PB5 pin
0x02: PC5 pin
0x03: PD5 pin
0x04: PE5 pin
0x05: PF5 pin
0x06: PG5 pin
0x07: PH5 pin
0x08: PI5 pin
0x09: PZ5 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
0x00: PA4 pin
0x01: PB4 pin
0x02: PC4 pin
0x03: PD4 pin
0x04: PE4 pin
0x05: PF4 pin
0x06: PG4 pin
0x07: PH4 pin
0x08: PI4 pin
0x09: PZ4 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

29.7.63 EXTI2 external interrupt selection register 3 (EXTI2_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
0x07: PH11 pin
0x08: PI11 pin
0x09: PZ11 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
0x00: PA10 pin
0x01: PB10 pin
0x02: PC10 pin
0x03: PD10 pin
0x04: PE10 pin
0x05: PF10 pin
0x06: PG10 pin
0x07: PH10 pin
0x08: PI10 pin
0x09: PZ10 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
0x00: PA9 pin
0x01: PB9 pin
0x02: PC9 pin
0x03: PD9 pin
0x04: PE9 pin
0x05: PF9 pin
0x06: PG9 pin
0x07: PH9 pin
0x08: PI9 pin
0x09: PZ9 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 7:0 **EXTI8[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
0x00: PA8 pin
0x01: PB8 pin
0x02: PC8 pin
0x03: PD8 pin
0x04: PE8 pin
0x05: PF8 pin
0x06: PG8 pin
0x07: PH8 pin
0x08: PI8 pin
0x09: PZ8 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

29.7.64 EXTI2 external interrupt selection register 4 (EXTI2_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
0x06: PG15 pin
0x07: PH15 pin
0x08: PI15 pin
0x09: PZ15 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
0x07: PH14 pin
0x08: PI14 pin
0x09: PZ14 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
0x03: PD13 pin
0x04: PE13 pin
0x05: PF13 pin
0x06: PG13 pin
0x07: PH13 pin
0x08: PI13 pin
0x09: PZ13 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
0x00: PA12 pin
0x01: PB12 pin
0x02: PC12 pin
0x03: PD12 pin
0x04: PE12 pin
0x05: PF12 pin
0x06: PG12 pin
0x07: PH12 pin
0x08: PI12 pin
0x09: PZ12 pin
Others: reserved

When SECx is disabled in EXTI2_SECCFGR1, this field can be accessed with non-secure and secure access.
When SECx is enabled in EXTI2_SECCFG1, this field can only be accessed with secure access. Non-secure write to this field is discarded. Non-secure read returns 0.
When PRIVx is disabled in EXTI2_PRIVCFGR1, this field can be accessed with unprivileged and privileged access.
When PRIVx is enabled in EXTI2_PRIVCFGR1, this field can only be accessed with privileged access. Unprivileged write to this field is discarded. Unprivileged read returns 0.
When CID filtering is disabled in EXTI2_ExCIDCFGR, this field can be accessed by any processor CID.
When CID filtering is enabled in EXTI2_ExCIDCFGR, this field can only be accessed by the allowed processor CID. Dis allowed CID write to this field is discarded, and read returns 0.

29.7.65 EXTI2 lock register (EXTI2_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register can only be written by the secure, privileged, trusted domain CID. A write access from any other protection level or domain CID is discarded and any read returns zero value. Both illegal write and read access generate an illegal access event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK:** Global security, privilege and CID configuration registers EXTI2_SECCFGRx, EXTI2_PRIVCFGRx, EXTI2_EnCIDCFGR and EXTI2_CmCIDCFGR lock

This bit is write once after reset.

0: Security and privilege configuration open, can be modified

1: Security and privilege configuration locked, can no longer be modified

29.7.66 EXTI2 CPUm wake-up with interrupt mask register (EXTI2_CmIMR1) (m = 1 to 2)

Address offset: Block 1: 0x080

Address offset: Block 2: 0x0C0

Reset value: 0x0000 0000

Contains register bits for configurable events and direct events.

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI2_CmCIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	Res.	Res.	IM26	Res.	IM24	IM23	IM22	IM21	Res.	IM19	IM18	IM17	IM16
rw	rw	rw			rw		rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 31 to 29)

0: Wake-up and interrupt CPU(m) from input event x is masked.

1: Wake-up and interrupt CPU(m) from input event x is unmasked. CPU(m) wake-up event and configurable event interrupt generated.

When SECx is disabled in EXTI2_SECCFGR1, IMx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, IMx can only be accessed with secure access. Non-secure write to IMx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, IMx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, IMx can only be accessed with privileged access. Unprivileged write to IMx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, IMx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, IMx can only be accessed by the allowed processor CID. Dis allowed CID write to IMx is discarded, and read returns 0.

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **IM26:** CPUm wake-up with interrupt mask on event input 26

Bit 25 Reserved, must be kept at reset value.

Bits 24:21 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 24 to 21)

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 19 to 0)

29.7.67 EXTI2 CPU2 wake-up with event mask register (EXTI2_C2EMR1)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI2_CmCIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EMx:** CPU(m) wake-up with event generation mask on event input x (x = 15 to 0)

0: Wake-up with event generation from line x is masked.

1: Wake-up with event generation from line x is unmasked.

When SECx is disabled in EXTI2_SECCFGR1, EMx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR1, EMx can only be accessed with secure access. Non-secure write to EMx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR1, EMx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR1, EMx can only be accessed with privileged access. Unprivileged write to EMx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, EMx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, EMx can only be accessed by the allowed processor CID. Dis allowed CID write to EMx is discarded, and read returns 0.

29.7.68 EXTI2 CPUm wake-up with interrupt mask register (EXTI2_CmIMR2) (m = 1 to 2)

Address offset: Block 1: 0x090

Address offset: Block 2: 0x0D0

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI2_CmCIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM53	IM52	IM51	IM50	IM49	Res.
rw										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	IM42	IM41	IM40	IM39	Res.	IM37	IM36	IM35	IM34	Res.	Res.
					rw	rw	rw	rw		rw	rw	rw	rw		

Bit 31 **IM63:** CPUm wake-up with interrupt mask on event input 63

0: Wake-up and interrupt CPU(m) from input event x is masked.

1: Wake-up and interrupt CPU(m) from input event x is unmasked. CPU(m) wake-up event and configurable event interrupt generated.

When SECx is disabled in EXTI2_SECCFGR2, IMx can be accessed with non-secure and secure access.

When SECx is enabled in EXTI2_SECCFGR2, IMx can only be accessed with secure access. Non-secure write to IMx is discarded. Non-secure read returns 0.

When PRIVx is disabled in EXTI2_PRIVCFGR2, IMx can be accessed with unprivileged and privileged access.

When PRIVx is enabled in EXTI2_PRIVCFGR2, IMx can only be accessed with privileged access. Unprivileged write to IMx is discarded. Unprivileged read returns 0.

When CID filtering is disabled in EXTI2_ExCIDCFGR, IMx can be accessed by any processor CID.

When CID filtering is enabled in EXTI2_ExCIDCFGR, IMx can only be accessed by the allowed processor CID. Dis allowed CID write to IMx is discarded, and read returns 0.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 53 to 49)

Bits 16:11 Reserved, must be kept at reset value.

Bits 10:7 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 42 to 39)

Bit 6 Reserved, must be kept at reset value.

Bits 5:2 **IMx:** CPUm wake-up with interrupt mask on event input x (x = 37 to 34)

Bits 1:0 Reserved, must be kept at reset value.

29.7.69 EXTI2 CPU2 wake-up with event mask register (EXTI2_C2EMR2)

Address offset: 0x0D4

Reset value: 0x0000 0000

This register has CID resource protection and can only be accessed by the allowed CID as defined in EXTI2_CmCIDCFGR. An illegal CID write access is discarded and a read access return zero data, also on both illegal accesses an illegal access event is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

29.7.70 EXTI2 event CID config register (EXTI2_EnCIDCFGR)

Valid range: n = 0 to 27, 29 to 31, 33 to 38, 40 to 44, 46 to 56, 59 to 70, 76

Address offset: 0x180 + 0x04 * n (n = 0 to 76)

Reset value: 0x0000 0000

This register can only be written by the secure, privileged, trusted domain CID. A write access from any other protection level or domain CID is discarded and any read returns zero value. Both illegal write and read access generate an illegal access event.

Write to these registers are ignored if GLOCK is enabled in EXTI2_LOCKR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]:** EXTI event n allowed CID value
If CID filtering is disabled in CFEN, these bits are ignored, else EXITI event n allowed processor CID number.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** EXTI event n CID filtering enable
0: Disable EXTI event x filtering (EXTI event x can be accessed by all CIDs).
1: Enable EXTI event x filtering (EXTI event x can only be accessed by the processor with CID = CPUm CID).

29.7.71 EXTI2 processor m CID configuration register (EXTI2_CmCIDCFGR)

Address offset: 0x300 + 0x04 * (m - 1) (m = 1 to 2)

Reset value: 0x0000 0000

This register can only be written by the secure, privilege, trusted domain CID. An illegal write access to this register generates an illegal access event.

Write to these registers are ignored if GLOCK is enabled in EXTI2_LOCKR.

This registers has no read access limitations, and can be read by any domain CID, and any access protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]:** CPUm CID domain identification
If CFEN = 0, these bits are ignored, and there is no CPUm CID filtering, else CPUm CID domain identification number.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN:** CID filtering enabled for CPUm EXTI2_CmIMRn and EXTI2_CmEMRn registers
0: Disable the CID filtering for CPUm.
1: Enable the CID filtering for CPUm.

29.7.72 EXTI2 hardware configuration registers 13 (EXTI2_HWCFGR13)

Address offset: 0x3C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[95:64]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged and no resource filtering event
1: Security, privilege and resource filtering capable event

29.7.73 EXTI2 hardware configuration registers 12 (EXTI2_HWCFGR12)

Address offset: 0x3C4

Reset value: 0x803E 07BE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[63:32]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged and no resource filtering event
1: Security, privilege and resource filtering capable event

29.7.74 EXTI2 hardware configuration registers 11 (EXTI2_HWCFGR11)

Address offset: 0x3C8

Reset value: 0xE5EF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SEC[31:0]:** Hardware configuration event security privilege and resource isolation capability
0: Non-secure, unprivileged and no resource filtering event
1: Security, privilege and resource filtering capable event

29.7.75 EXTI2 hardware configuration registers 7 (EXTI2_HWCFGR7)

Address offset: 0x3D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[95:64]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.76 EXTI2 hardware configuration registers 6 (EXTI2_HWCFGR6)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[63:32]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.77 EXTI2 hardware configuration registers 5 (EXTI2_HWCFGR5)

Address offset: 0x3E0

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[31:0]:** Hardware configuration CPU event generation
0: No CPU event generated
1: CPU event generated

29.7.78 EXTI2 hardware configuration registers 4 (EXTI2_HWCFGR4)

Address offset: 0x3E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[95:64]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.79 EXTI2 hardware configuration registers 3 (EXTI2_HWCFGR3)

Address offset: 0x3E8

Reset value: 0x0038 003C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[63:32]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.80 EXTI2 hardware configuration registers 2 (EXTI2_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT_TRG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[31:0]:** Hardware configuration event trigger type
0: Direct event
1: Configurable event

29.7.81 EXTI2 hardware configuration register 1 (EXTI2_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0309 2141

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CIDWIDTH[3:0]				NBIOPORT[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUEVTEN[3:0]				NBCPUS[3:0]				NBEVENTS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CIDWIDTH[3:0]:** CID parameters bit width

Bits 23:16 **NBIOPORT[7:0]:** Hardware configuration of number of I/O ports on EXTI (n+1)

Bits 15:12 **CPUEVTEN[3:0]:** Hardware configuration of CPU(m) event output enable.

Bits 11:8 **NBCPUS[3:0]:** Hardware configuration number of CPUs (n+1)

Bits 7:0 **NBEVENTS[7:0]:** Hardware configuration number of event (n+1)

29.7.82 EXTI2 version register (EXTI2_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision number

Bits 3:0 **MINREV[3:0]:** Minor revision number

29.7.83 EXTI2 identification register (EXTI2_IPIDR)

Address offset: 0x3F8

Reset value: 0x000E 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]:** EXTI2 identification

29.7.84 EXTI2 size identification register (EXTI2_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

29.7.85 EXTI1 register map

Table 203. EXTI1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**EXTI1_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**EXTI1_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**EXTI1_SWIER1**	Res	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	SWI[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**EXTI1_RPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**EXTI1_FPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**EXTI1_ SECCFGR1**	SEC[31:26]						Res.	Res.	SEC [23:21]			Res.	SEC[19:0]																			
	Reset value	0	0	0	0	0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**EXTI1_ PRIVCFGR1**	PRIV[31:26]						Res.	Res.	PRIV [23:21]			Res.	PRIV[19:0]																			
	Reset value	0	0	0	0	0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved	Reserved																													

Table 203. EXTI1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	**EXTI1_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT44	RT43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0											
0x024	**EXTI1_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT44	FT43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0											
0x028	**EXTI1_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI44	SWI43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0											
0x02C	**EXTI1_RPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF44	RPIF43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0											
0x030	**EXTI1_FPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF44	FPIF43	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0											
0x034	**EXTI1_ SECCFGR2**	Res.	SEC[62:59]				Res.	SEC[57:52]						Res.	Res.	Res.	SEC [48:46			Res.	SEC [44:43]		Res.	SEC[41:36]						Res.	Res.	Res.	SEC32
	Reset value		0	0	0	0		0	0	0	0	0	0				0	0	0		0	0		0	0	0	0	0	0				0
0x038	**EXTI1_ PRIVCFGR2**	Res.	PRIV[62:59]				Res.	PRIV[57:52]						Res.	Res.	Res.	PRIV [48:46]			Res.	PRIV [44:43]		Res.	PRIV[41:36]						Res.	Res.	Res.	PRIV32
	Reset value		0	0	0	0		0	0	0	0	0	0				0	0	0		0	0		0	0	0	0	0	0				0
0x03C	Reserved	Reserved																															
0x040	**EXTI1_RTSR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT73	Res.	Res.	Res.	Res.	Res.	RT67	Res.	RT65	RT64
	Reset value																							0						0		0	0
0x044	**EXTI1_FTSR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT73	Res.	Res.	Res.	Res.	Res.	FT67	Res.	FT65	FT64
	Reset value																							0						0		0	0
0x048	**EXTI1_SWIER3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI73	Res.	Res.	Res.	Res.	Res.	SWI67	Res.	SWI65	SWI64
	Reset value																							0						0		0	0
0x04C	**EXTI1_RPR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF73	Res.	Res.	Res.	Res.	Res.	RPIF67	Res.	RPIF65	RPIF64
	Reset value																							0						0		0	0
0x050	**EXTI1_FPR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF73	Res.	Res.	Res.	Res.	Res.	FPIF67	Res.	FPIF65	FPIF64
	Reset value																							0						0		0	0
0x054	**EXTI1_ SECCFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[82:72]												Res.	SEC70	Res.	SEC [68:67]		Res.	SEC65	SEC64
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0		0		0	0		0	0

Table 203. EXTI1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	**EXTI1_ PRIVCFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[82:72]											Res.	PRIV70	Res.	PRIV [68:67]		Res.	PRIV65	PRIV64
	Reset value														0	0	0	0	0	0	0	0	0	0	0		0		0	0		0	0
0x05C	Reserved	Reserved																															
0x060	**EXTI1_EXTICR1**	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**EXTI1_EXTICR2**	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	**EXTI1_EXTICR3**	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	**EXTI1_EXTICR4**	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI2[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**EXTI1_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x074-0x07C	Reserved	Reserved																															
0x080	**EXTI1_C1IMR1**	IM[31:26]						Res.	Res.	IM[23:21]			Res.	IM[19:0]																			
	Reset value	0	0	0	0	0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084-0x08C	Reserved	Reserved																															
0x090	**EXTI1_C1IMR2**	Res.	IM[62:59]				Res.	IM[57:52]						Res.	Res.	Res.	IM[48:46]		Res.	IM [44: 43]		Res.	IM[41:36]						Res.	Res.	Res.	IM32	
	Reset value		0	0	0	0		0	0	0	0	0	0				0	0	0		0	0		0	0	0	0	0	0				0
0x094-0x09C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0A0	**EXTI1_IMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[82:72]											Res.	IM70	Res.	IM[68:67]		Res.	IM65	IM64
	Reset value														0	0	0	0	0	0	0	0	0	0	0		0		0	0		0	0
0x0A4 - 0x0BC	Reserved	Reserved																															
0x0C0	**EXTI1_C2IMR1**	IM[31:26]						Res.	Res.	IM[23:21]			Res.	IM[19:0]																			
	Reset value	0	0	0	0	0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 203. EXTI1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C4	**EXTI1_C2EMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved	Reserved																															
0x0D0	**EXTI1_C2IMR2**	Res.	IM[62:59]				Res.	IM[57:52]						Res.	Res.	Res.	IM[48:46]		Res.	IM[44:43]		Res.	IM[41:36]							Res.	Res.	Res.	IM32
	Reset value		0	0	0	0		0	0	0	0	0	0				0	0	0		0	0		0	0	0	0	0	0				0
0x0D4-0x0DC	Reserved	Reserved																															
0x0E0	**EXTI1_C2IMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[82:72]											Res.	IM70	Res.	IM[68:67]		Res.	IM65	IM64
	Reset value														0	0	0	0	0	0	0	0	0	0	0		0		0	0		0	0
0x0E4	**EXTI1_C2EMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM65	Res.
	Reset value																															0	
0x0E8 - 0x17C	Reserved	Reserved																															
0x180 + 0x04*n (n=0 to 84) (1)	**EXTI1_EnCIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x1D0	Reserved	Reserved																															
0x300	**EXTI1_C1CIDCFRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x304	**EXTI1_C2CIDCFRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x308-0x3BC	Reserved	Reserved																															
0x3C0	**EXTI1_HWCFGR13**	SEC[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1
0x3C4	**EXTI1_HWCFGR12**	SEC[63:32]																															
	Reset value	0	1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1	0	1	1	1	1	1	1	0	0	0	1
0x3C8	**EXTI1_HWCFGR11**	SEC[31:0]																															
	Reset value	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3CC-0x3D4	Reserved	Reserved																															

Table 203. EXTI1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3D8	**EXTI1_ HWCFGR7**	CPUEVENT[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3DC	**EXTI1_ HWCFGR6**	CPUEVENT[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E0	**EXTI1_ HWCFGR5**	CPUEVENT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3E4	**EXTI1_ HWCFGR4**	EVENT_TRG[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	
0x3E8	**EXTI1_ HWCFGR3**	EVENT_TRG[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0x3EC	**EXTI1_ HWCFGR2**	EVENT_TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3F0	**EXTI1_ HWCFGR1**	Res.	Res.	Res.	Res.	CIDWIDTH				NBIOPORT[7:0]								CPUEVTEN [3:0]				NBCPUS[3: 0]				NBEVENTS[7:0]							
	Reset value					0	0	1	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0x3F4	**EXTI1_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3: 0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	0	1
0x3F8	**EXTI1_IPIDR**	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	**EXTI1_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

1. Valid discontinuous range: n = 0 to 27, 29 to 31, 33 to 38, 40 to 44, 46 to 56, 59 to 70, 76.

Refer to *Section 2.4* for the register boundary addresses.

29.7.86 EXTI2 register map

Table 204. EXTI2 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**EXTI2_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**EXTI2_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**EXTI2_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**EXTI2_RPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**EXTI2_FPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**EXTI2_ SECCFGR1**	SEC [31:29]			Res.	Res.	SEC26	Res.	SEC[24:21]				Res.	SEC[19:0]																			
	Reset value	0	0	0			0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**EXTI2_ PRIVCFGR1**	PRIV [31:29]			Res.	Res.	PRIV26	Res.	PRIV[24:21]				Res.	PRIV[19:0]																			
	Reset value	0	0	0			0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved	Reserved																															
0x020	**EXTI2_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[53:51]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[37:34]			Res.	Res.
	Reset value											0	0	0															0	0	0	0		
0x024	**EXTI2_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[53:51]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[37:34]			Res.	Res.
	Reset value											0	0	0															0	0	0	0		
0x028	**EXTI2_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI [53:51]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI[37:34]			Res.	Res.
	Reset value											0	0	0															0	0	0	0		
0x02C	**EXTI2_RPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF [53:51]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF[37:34]			Res.	Res.
	Reset value											0	0	0															0	0	0	0		
0x030	**EXTI2_FPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF [53:51]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF[37:34]			Res.	Res.
	Reset value											0	0	0															0	0	0	0		

Table 204. EXTI2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	**EXTI2_ SECCFGR2**	SEC63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[53:49]					Res.	Res.	Res.	Res.	Res.	Res.	SEC[42:39]				Res.	SEC[37:34]				Res.	Res.
	Reset value	0										0	0	0	0	0							0	0	0	0		0	0	0	0		
0x038	**EXTI2_ PRIVCFGR2**	PRIV63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[53:49]					Res.	Res.	Res.	Res.	Res.	Res.	PRIV[42:39]				Res.	PRIV[37:34]				Res.	Res.
	Reset value	0										0	0	0	0	0							0	0	0	0		0	0	0	0		
0x03C-0x05C	Reserved	Reserved																															
0x060	**EXTI2_ EXTICR1**	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]										EXTI0[7:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**EXTI2_ EXTICR2**	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]										EXTI4[7:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	**EXTI2_ EXTICR3**	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]										EXTI8[7:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	**EXTI2_ EXTICR4**	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]										EXTI2[7:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**EXTI2_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x074-0x07C	Reserved	Reserved																															
0x080	**EXTI2_C1IMR1**	IM[31:29].			Res.	Res.	IM26	Res.	IM[24:21].				Res.	IM[19:0].																			
	Reset value	0	0	0			0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084-0x08C	Reserved	Reserved																															
0x090	**EXTI2_C1IMR2**	IM63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[53:49]					Res.	Res.	Res.	Res.	Res.	Res.	IM[42:39]				Res.	IM[37:34]				Res.	Res.
	Reset value	0										0	0	0	0	0							0	0	0	0		0	0	0	0		
0x094 - 0x0BC	Reserved	Reserved																															
0x0C0	**EXTI2_C2IMR1**	IM[31:29].			Res.	Res.	IM26	Res.	IM[24:21].				Res.	IM[19:0].																			
	Reset value	0	0	0			0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	**EXTI2_ C2EMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 204. EXTI2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C8-0x0CC	Reserved	Reserved																															
0x0D0	**EXTI2_C2IMR2**	IM63	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[53:49]					Res.	Res.	Res.	Res.	Res.	Res.	IM[42:39]				Res.	IM[37:34]				Res.	Res.
	Reset value	0										0	0	0	0	0							0	0	0	0		0	0	0	0		
0x0D4	**EXTI2_ C2EMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0D8-0x17C	Reserved	Reserved																															
0x180 + 0.04*n (n=0 to 76) (1)	**EXTI2_ EnCIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x2B4-0x2FC	Reserved	Reserved																															
0x300	**EXTI2_ C1CIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x304	**EXTI2_ C2CIDCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x308-0x3BC	Reserved	Reserved																															
0x3C0	**EXTI2_ HWCFGR13**	SEC[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C4	**EXTI2_ HWCFGR12**	SEC[63:32]																															
	Reset value	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0
0x3C8	**EXTI2_ HWCFGR11**	SEC[31:0]																															
	Reset value	1	1	1	0	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3CC-0x3D4	Reserved	Reserved																															
0x3D8	**EXTI2_ HWCFGR7**	CPUEVENT[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3DC	**EXTI2_ HWCFGR6**	CPUEVENT[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 204. EXTI2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3E0	**EXTI2_ HWCFGR5**	CPUEVENT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3E4	**EXTI2_ HWCFGR4**	EVENT_TRG[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E8	**EXTI2_ HWCFGR3**	EVENT_TRG[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
0x3EC	**EXTI2_ HWCFGR2**	EVENT_TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3F0	**EXTI2_ HWCFGR1**	Res.	Res.	Res.	Res.	CIDWIDTH				NBIOPORT[7:0]								CPUEVTEN [3:0]				NBCPUS[3: 0]				NBEVENTS[7:0]							
	Reset value					0	0	1	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1
0x3F4	**EXTI2_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3: 0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	0	1
0x3F8	**EXTI2_IPIDR**	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	**EXTI2_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

1. Valid discontinuous range: n = 0 to 19, 21 to 34, 36 to 50, 52 to 57, 59 to 62, 64 to 65, 67 to 68, 70, 72 to 84.

Refer to *Section 2.4* for the register boundary addresses.

30 Cyclic redundancy check calculation unit (CRC)

30.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

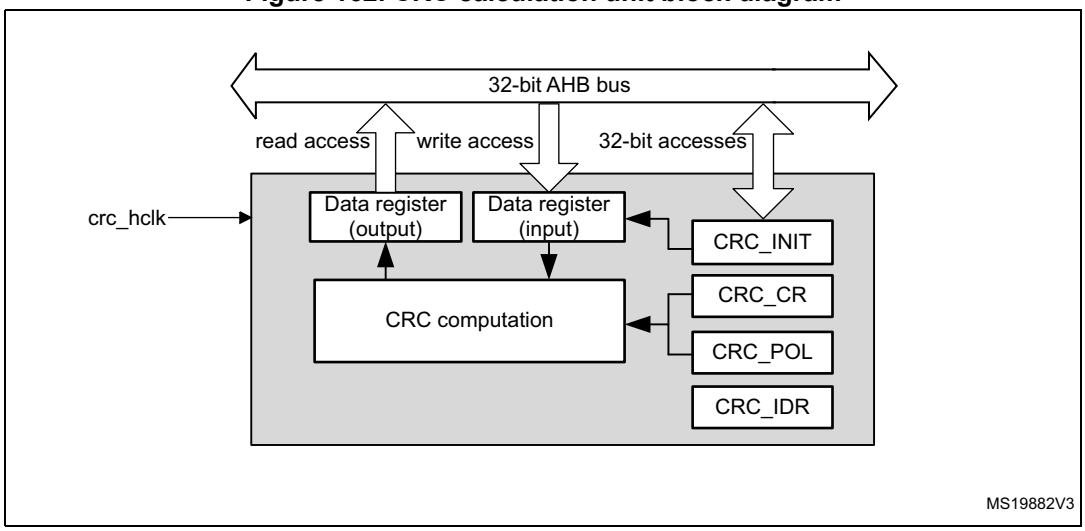
30.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

30.3 CRC functional description

30.3.1 CRC block diagram

Figure 162. CRC calculation unit block diagram



30.3.2 CRC internal signals

Table 205. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

30.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

30.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

30.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

30.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

30.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
					rw	rw	rw	rw	rw	rw	rw	rw			rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

30.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

30.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

30.4.6 CRC register map

Table 206. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

31 Mass storage interfaces

31.1 Overview

STM32MP21xx devices embed three types of mass storage interfaces:

- FMC interface: manages access to external parallel NOR, parallel NAND, or parallel static RAM devices.
- Octo-SPI interface: manages access to external serial NOR or serial NAND devices.
- SD/*e*.MMC interface: manages access to external SD or *e*.MMC or SDIO devices.

31.2 FMC interface

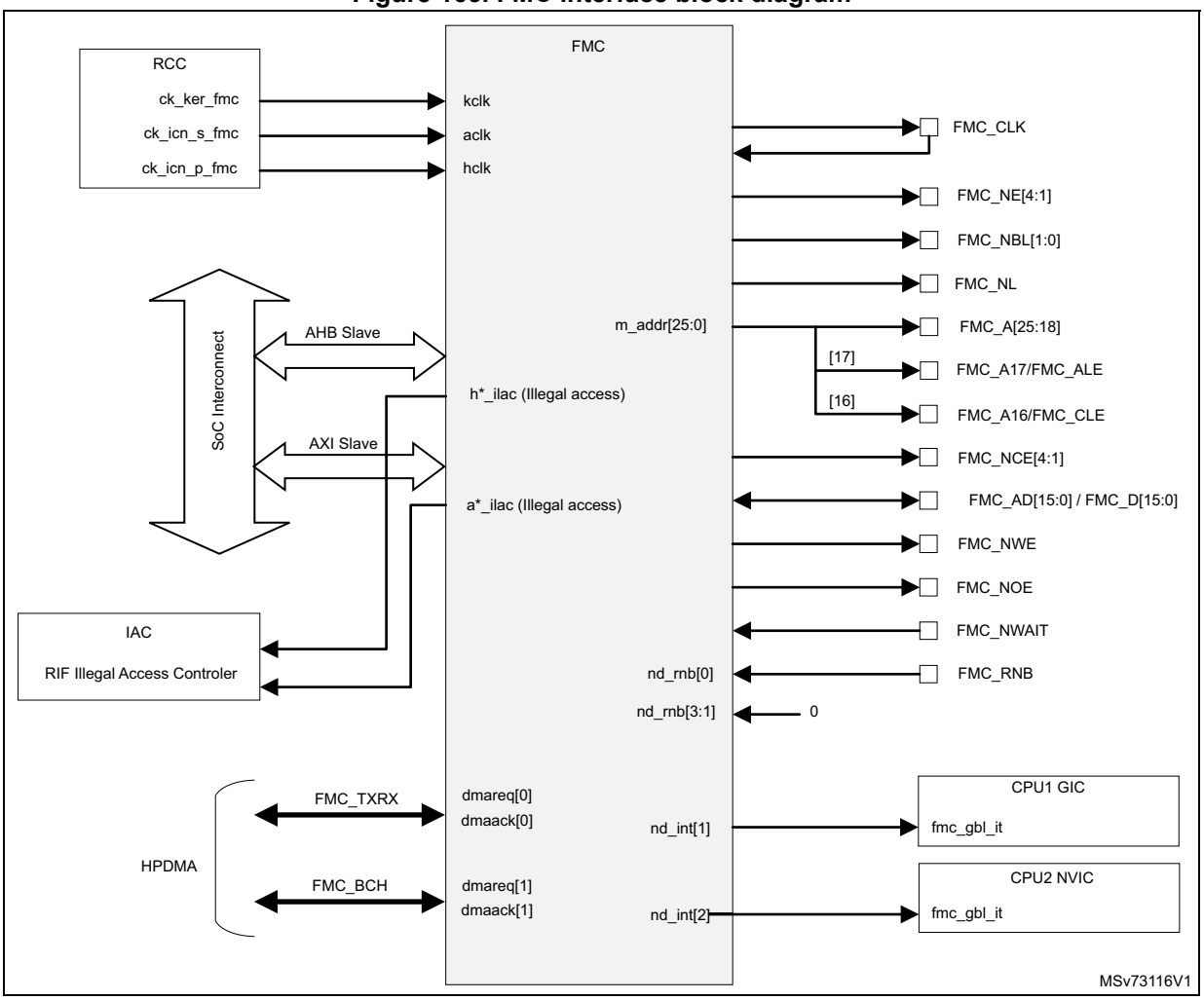
The FMC is compliant with the RIF (resource isolation framework). See *Section 32: Flexible memory controller (FMC)*.

Main features of the FMC interface

- NOR/PSRAM:
  - ‘Generic interface’ to parallel NOR/PSRAM or external device 8080 display or external FPGA
  - Parallel NOR flash memory not supported as BOOT device
  - Up to 4 regions (4 CS, 64 Mbytes each), independent memory and RIF configuration for each region
  - 8- or 16-bit data modes
  - Separate or multiplexed address and data buses
  - A clock divided from the kernel clock provided for synchronous devices
- NAND:
  - SLC NAND ONFI 1.0
  - Hamming and BCH ECC controllers
  - Supported as BOOT device
  - 4 CS + 1 RnB signals
  - 8- or 16-bit data modes
  - Hardware sequencer which can be used jointly with DMA channels to free the CPU of sequencing tasks
  - RIF compliant

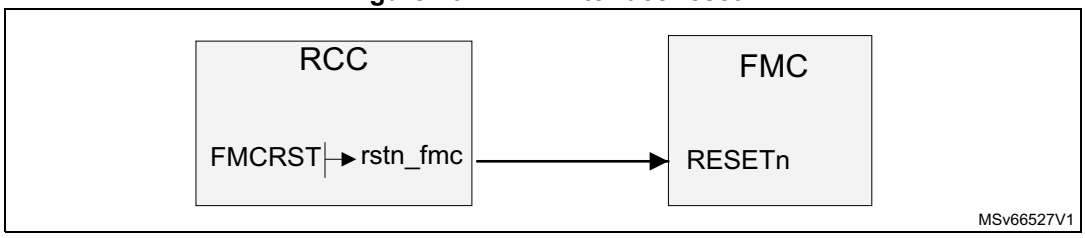
31.2.1 FMC interface block diagram

Figure 163. FMC interface block diagram



31.2.2 FMC interface reset

Figure 164. FMC interface reset



There is only one reset for the FMC interface (rstn_fmc) which is controlled by FMCRST in RCC_FMCCFGR register.

31.2.3 FMC interface interrupt

The FMC generates two interrupts, one for each CID that can be attached to the NAND controller:

- nd_int[1] generated if the CID attached to the NAND controller corresponds to CPU1
- nd_int[2] generated if the CID attached to the NAND controller corresponds to CPU2

Table 207. FMC interface interrupts

FMC signal name	CPU1 interrupt	CPU2 interrupt
nd_int[1]	fmc_gbl_it	-
nd_int[2]	-	fmc_gbl_it

31.3 Octo-SPI interface

The Octo-SPI interface offers one OCTOSPI physical port. Accesses to the OCTOSPI controller are protected by RISUP components.

31.3.1 Main features of the Octo-SPI interface

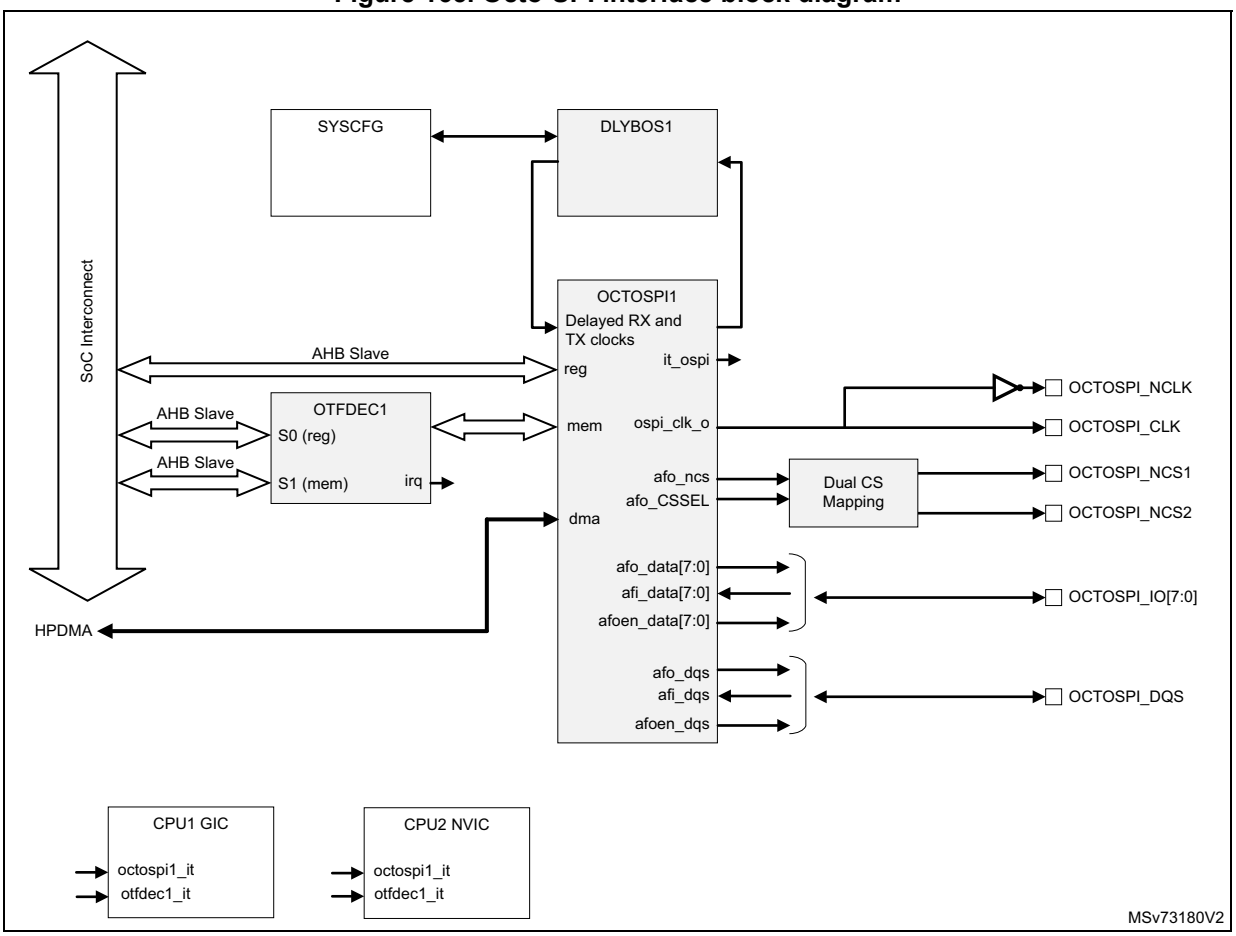
- Support sNOR and sNAND devices from different families: HyperBus™ compliant devices (HyperRAM™ and HyperFlash™ devices from Spansion, Cypress, including MCP devices which integrate both HyperRAM™ and HyperFlash™), Micron devices (Xcella family), Macronix™ devices.
- Three functional modes: indirect, automatic-status polling, memory-mapped
- Up to 4 Gbytes of external memory can be addressed in indirect mode (per physical port and per CS), and up to 256 Mbytes in memory-mapped mode (combined for both physical ports and per CS).
- Single-, dual-, quad-, and octal-SPI communication
- Dual-quad communication
- Single data rate (SDR) and double transfer rate (DTR)
- Maximum target frequency is 133 MHz for SDR and 133 MHz for DTR
- Data strobe support
- DMA channel for indirect mode
- Support for on-the-fly decryption of data read from an external flash memory:
  - Up to four regions can be defined with independent decryption attributes (execute-only and execute-never) and keys.
  - The decryption applies when the OCTOSPI controller is used in read memory-mapped mode.
- Encryption support
- Double CS mapping that can be used to address two external flash memory devices with a single OCTOSPI controller.

For a complete list of features, refer to:

- *Section 33: Octo-SPI interface (OCTOSPI)*
- *Section 51: On-the-fly decryption engine (OTFDEC)*

31.3.2 Octo-SPI interface block diagram

Figure 165. Octo-SPI interface block diagram



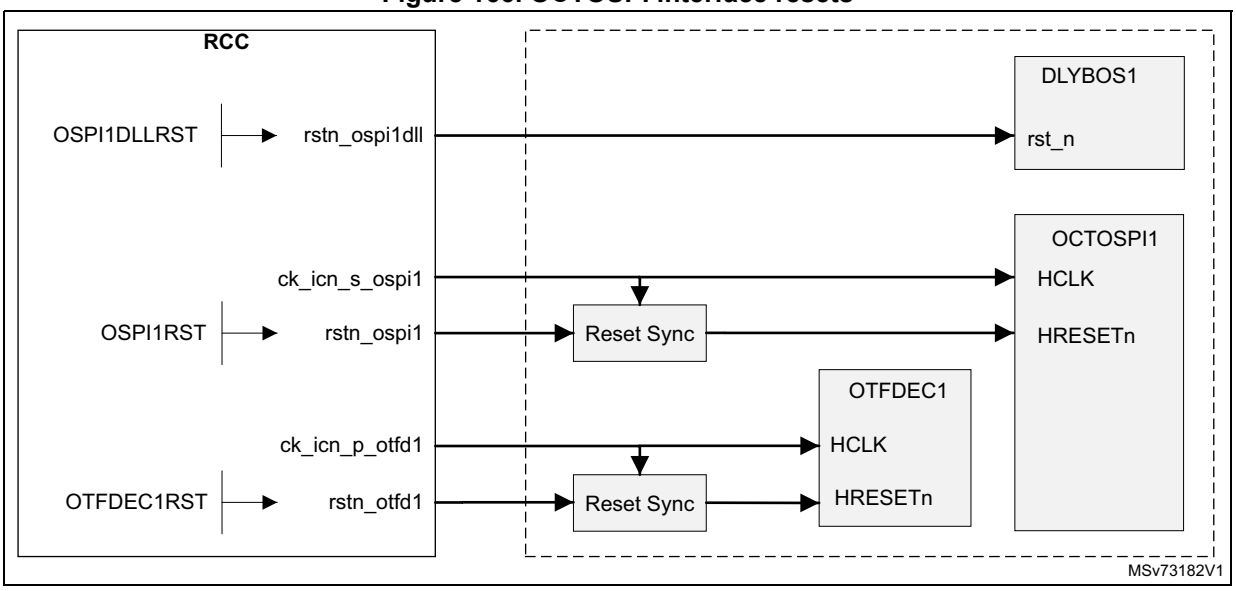
OCTOSPI1 is the memory controller. It can be accessed by either CPU1 or CPU2. It provides 1 chip select which can be mapped on OCTOSPI_NCS1 or OCTOSPI_NCS2 depending on CSSEL setting.

OTFDEC1 is in charge of on-the-fly decryption of data read from the external memory devices.

DLYBOS1 is a DLL-based programmable delay which allows to tune the phase of the RX sampling clock and the TX launch clock.

31.3.3 Octo-SPI interface resets

Figure 166. OCTOSPI interface resets



There are three independent resets for the OCTOSPI interface:

- rstn_ospi1dll which is controlled by the OSPI1DLLRST bit in RCC_OSPI1CFGR register.
- rstn_ospi1 which is controlled by the OSPI1RST bit in RCC_OSPI1CFGR register.
- rstn_otfd1 which is controlled by the OTFDEC1RST bit in RCC_OSPI1CFGR register.

When configuring the OCTOSPI interface, the reset sequence to be applied is:

1. At device reset, OSPI1DLLRST, OSPI1RST, OTFDEC1RST are all 0. DLYBOS1, OCTOSPI1, OTFDEC1 are under reset because their clocks are disabled.
2. Set OSPI1EN. This activates the peripheral clock of OCTOSPI1 and OTFDEC1 and releases their resets.
3. Configure the OCTOSPI1 controller and OTFDEC1 engine.
4. For configurations where the programmable delays must be used:
   a) Wait for the DLL to be locked. This is indicated by the bit LOCK in SYSCFG_DLYBOS1SR register.
   b) Perform a training sequence to determine which delay values correspond to the middle of the valid window for RX sampling (and possibly Tx launching)

For configurations where the programmable delay is not needed, DLYBOS1 can be maintained under reset by setting OSPI1DLLRST to 1.

31.3.4 Octo-SPI interface memory

256 MBytes are allocated in STM32MP21xx memory map for OCTOSPI1. This area is used when OCTOSPI1 is operating in memory-mapped mode.

31.3.5 OCTOSPI1 description

Refer to *Section 33: Octo-SPI interface (OCTOSPI)*.

31.3.6 OTFDEC1 description

Refer to *Section 51: On-the-fly decryption engine (OTFDEC)*.

31.3.7 DLYBOS1 description

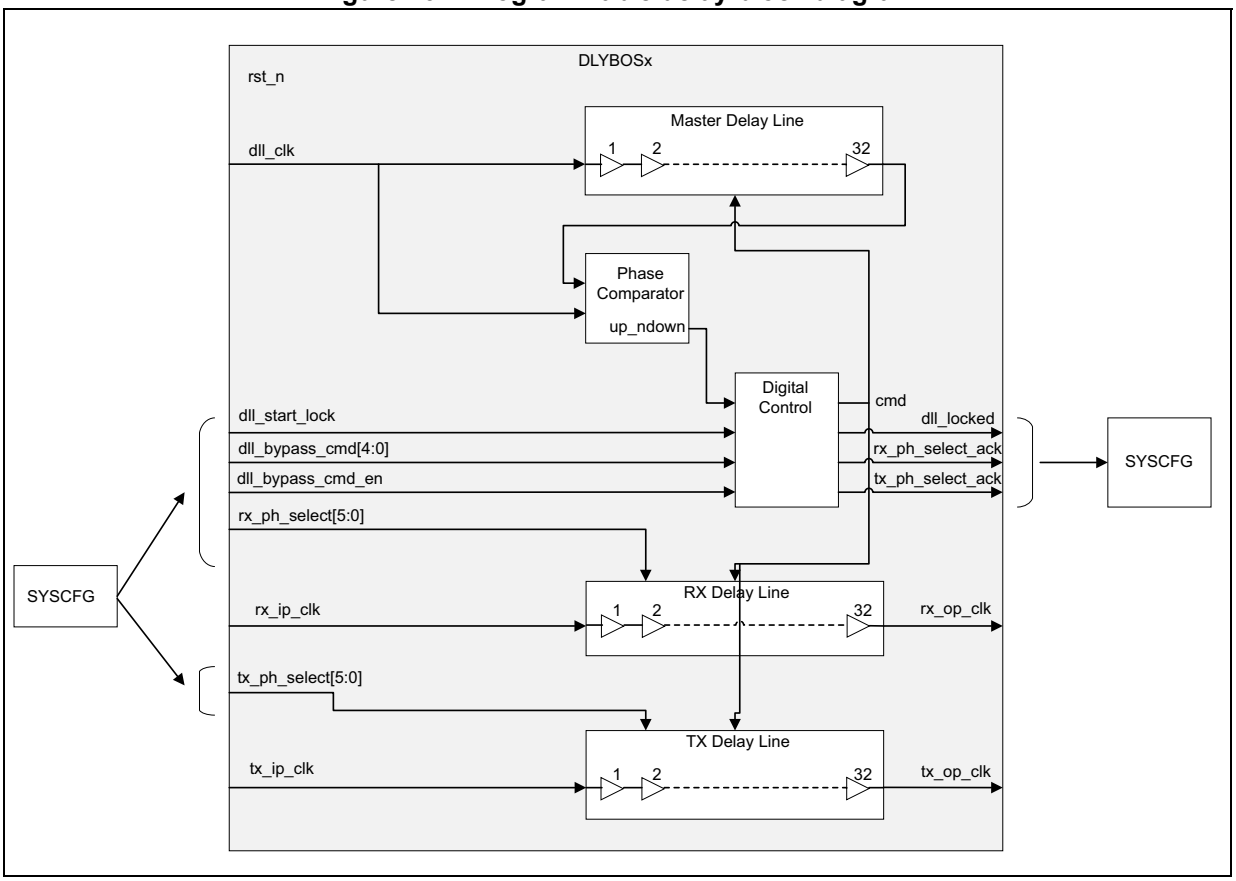
Refer to *Section 35: Delay block (DLYB)*

The programmable delay is used to tune the phase of:

- the RX sampling clock (or DQS) in order to sample the data in their valid window
- the TX launch clock in order to meet setup and hold constraints of TX signals versus the memory clock

The programmable delay block is based on a DLL that compensates for voltage and temperature variations. The DLL features 32 taps and covers the [50 MHz, 208 MHz] frequency range. The clock on which the DLL locks is a free running clock provided by the OCTOSPI. This clock is at the same frequency as the clock sent to the memory. When OCTOSPI1 interface operates below 50 MHz, either the programmable delay is not used (standard case), or it must be used in bypass mode (case of OCTOSPI devices with DQS strobe).

Figure 167. Programmable delay block diagram



The DLL architecture is based on one master delay line, and two slave delay lines (RX and TX). There are two operating modes:

- Lock mode (dll_bypass_cmd_en = 0)
  - This mode can only be used when the dll_clk frequency is in the range [50 MHz, 208 MHz].
  - The DLL is enabled by setting dll_start_lock to 1.
  - The DLL locks on the frequency of the dll_clk input clock, and compensates for voltage and temperature variations. This process determines the command to be applied to each of the 32 delay units (named tap) of the master delay line for the total master delay to represent one period of dll_clk. This command is copied to each delay unit of the RX and TX delay lines. The command varies with voltage and temperature for the DLL to remain locked. This guarantees that each delay unit of the RX and TX delay lines represents 1/32th of the input clock period.
  - Independent delay values can be selected for the RX and TX delay lines by controlling rx_ph_select[5:0] and tx_ph_select[5:0] parameters independently. The granularity of the delay is 1/32th of the input clock period.

Table 208. RX and TX delays in lock mode

rx_ph_select[5:0] (respectively tx_ph_select[5:0])	RX delay (respectively TX delay)
000000	0
000001	1/32 x T(dll_clk)
...	...
011111	31/32 x T(dll_clk)
100000	32/32 x T(dll_clk)

  - When selecting a new delay value with rx_ph_select[5:0] (respectively tx_ph_select[5:0]), an acknowledge is returned when the change takes place with rx_ph_select_ack (respectively tx_ph_select_ack).
- Bypass mode (dll_bypass_cmd_en = 1)
  - In this mode, delays in RX and TX delay lines are used as “pure” delays: without trying to lock on the frequency of the dll_clk input clock. There is therefore no voltage nor temperature compensation in this mode, and RX and TX delay values vary with the voltage and the temperature. The command applied to each delay unit of the RX and TX delay lines is controlled by dll_bypass_cmd[4:0]. Each delay unit includes 24 delay elements. The number of delay elements which are enabled in each delay unit depends on the value of dll_bypass_cmd[4:0].

Table 209. Number of delay elements enabled in each delay unit

dll_bypass_cmd[4:0]	Number of delay elements enabled
00000	reserved
00001	1 delay element is enabled
00010	2 delay elements are enabled
...	...
11000 to 11111	All 24 delay elements are enabled

- Independent delay values can be selected for the RX and TX delay lines by controlling rx_ph_select[5:0] and tx_ph_select[5:0] parameters independently. Rx_ph_select[5:0] and tx_ph_select[5:0] control the number of delay units that are enabled in the RX and TX delay lines: a value of n corresponds to n delay units enabled. The total delay inserted by the RX (respectively TX) delay line depends on rx_ph_select[5:0] (respectively tx_ph_select[5:0]) and dll_bypass_cmd[4:0] values.
- This mode is required to be able to insert a delay between the rx_ip_clk (respectively tx_ip_clk) input clock and the rx_op_clk (respectively tx_op_clk) output clock when the operating frequency is lower than the minimum frequency supported by the DLL (f <50 MHz). This is necessary to support OCTOSPI devices in DTR mode at low frequency for which DQS must be delayed to sample data.

31.3.8 Octo-SPI interface interrupts

OCTOPSI1 and OTFDEC1 generate one interrupt.

Table 210. Octo-SPI interface interrupts

Peripheral	Signal name	CPU1 interrupt	CPU2 interrupt
OCTOSPI1	it_ospi	octospi1_it	octospi1_it
OTFDEC1	irq	otfdec1_it	otfdec1_it

31.3.9 Chip select management

The chip select signal provided by the OCTOSPI controller is mapped on OCTOSPI_NCS1 or OCTOSPI_NCS2 depending on the value of CSSEL bit in OCTOSPI_CR register.

Table 211. OCTOSPI1 CS mapping

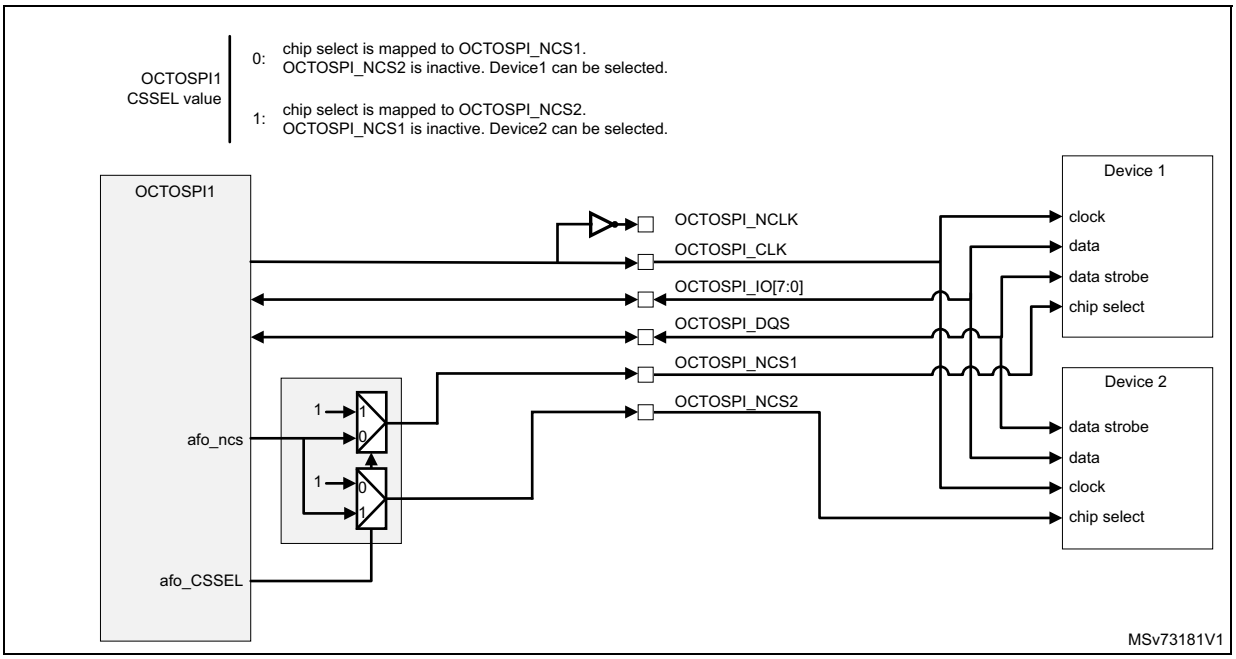
OCTOSPI1 CSSEL	OCTOSPI1 CS mapped on
0	OCTOSPI_NCS1
1	OCTOSPI_NCS2

Note: It is possible to connect 2 flash devices to the OCTOSPI interface and control them both by the OCTOSPI controller. However, the two devices cannot be used simultaneously. Switching from one device to the other requires to:

- *re-program OCTOSPI1 CSSEL to remap the chip select.*
- *re-program the OCTOSPI controller.*

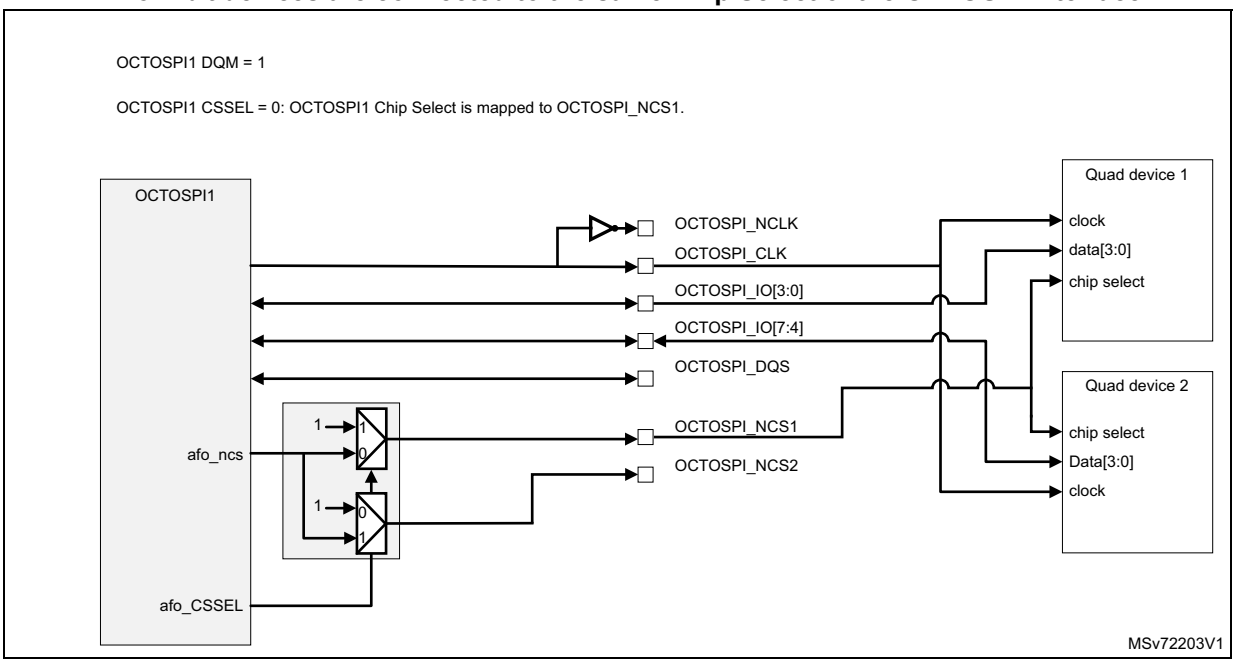
The figure below gives an example of this possibility where two 8-bit devices are attached to the OCTOSPI interface.

Figure 168. Flash configuration example:
2× 8-bit flash devices connected to the OCTOSPI interface



The OCTOSPI interface also supports dual-quad mode. In dual quad mode, two 4-bit devices are connected to the same chip select in order to provide a throughput similar to an 8-bit device. This mode is configured by the bit DQM in OCTOSPI_CR register. An example is shown in the figure below.

Figure 169. Flash configuration example: dual quad mode.
Two 4-bit devices are connected to the same Chip Select of the OCTOSPI interface



31.3.10 Octo-SPI interface programmable parameters

Octo-SPI memory area allocation in memory-mapped mode

Refer to *Section 31.3.4: Octo-SPI interface memory*.

DLYBOS1 control

The DLYBOS1 programmable delay is controlled by bits in SYSCFG_DLYBOS1CR and SYSCFG_DLYBOS1SR registers. The table below gives the correspondence between SYSCFG bitfields and the control parameters described in *Section 31.3.7*.

Table 212. DLYBOS1 SYSCFG bits

DLYBOS1 SYSCFG bits	Control input or status output in the OCTOSPI	Comment
BYP_EN	dll_bypass_cmd_en	Selection of lock or pure-delay mode
EN	dll_start_lock	Enable of DLL when configured in Lock mode
RX_TAP_SEL[5:0]	rx_ph_select[5:0]	Selection of delay in RX delay line
TX_TAP_SEL[5:0]	tx_ph_select[5:0]	Selection of delay in TX delay line
BYP_CMD[4:0]	dll_bypass_cmd[4:0]	Command applied to every delay units in RX and TX delay lines when the DLL is configured in pure-delay mode
LOCK	dll_locked	When in lock mode, indicates that the DLL has locked on the input clock frequency

Table 212. DLYBOS1 SYSCFG bits (continued)

DLYBOS1 SYSCFG bits	Control input or status output in the OCTOSPI	Comment
RX_TAP_SEL_ACK	rx_ph_select_ack	Indicates that the RX delay line has switched to the delay selected by RX_TAP_SEL[5:0]
TX_TAP_SEL_ACK	tx_ph_select_ack	Indicates that the TX delay line has switched to the delay selected by TX_TAP_SEL[5:0]

31.3.11 Select the RX sampling strobe and configure the TX clock

The table below lists the various OCTOSPI modes, and gives guidelines to select the RX sampling strobe and configure the TX clock. Some information is also given on how to enable the DLL.

Table 213. OCTOSPI modes and recommended clock configuration

OCTOSPI Mode	STR or DTR	DQS used in read ?	Max frequency	Setup/Hold constraints on memory device	Data valid Window at output of memory Device	TX clock configuration	RX sampling strobe selection
HyperBus(Cypress, Spansion)	DTR	yes	200 166 133 100 50	0.40 / 0.40 ns 0.60 / 0.60 ns 0.80 / 0.80 ns 1 / 1 ns wrt memory clock edges	1.45 ns 1.70 ns 2.37 ns 3.30 ns	if kernel clock is a multiple of memory clock, set DHQC = 1 (this automatically bypasses the TX delay). If kernel clock is equal to memory clock, set DQHC = 0. The DLL must be used for TX delay.	DQS delayed (DQSE = 1, DLYBYP = 0). The DLL must be used for RX delay.
Macronix [For instr. READ, FASTREAD, OCTAREAD-STR]	STR	no	133 100 66	2 / 2 ns 2 / 2 ns 2 / 2 ns wrt memory clock rising edge	2 ns@10pF 4 ns@15pF 7 ns@30pF	Set DQHC = 0 and configure the TX delay to minimum value.	Feedback clock delayed @133 MHz and 100 MHz (DQSE=0, DLYBYP=0). The DLL must be used for RX delay. Feedback clock @66 MHz (DQSE=0, DLYBYP=1). The DLL is not used for RX delay.
Macronix [For instr. OCTAREAD - DTR]	DTR	yes	133 100 66	0.8 / 0.8 ns 1 / 1 ns 1 / 1 ns	1.5ns@10pF 2.8ns @15pF 4.9ns @30pF	if kernel clock is a multiple of memory clock, set DHQC = 1 (this automatically bypasses the TX delay). If kernel clock is equal to memory clock, set DQHC = 0. The DLL must be used for TX delay.	DQS delayed (DQSE = 1, DLYBYP = 0). The DLL must be used for RX delay.
Micron Xcella (MT35XL01 GBBA ds)	STR	no	133	1.8 / 1.8 ns Wrt memory clock rising edge	2.8ns @12pF	Set DQHC = 0 and configure the TX delay to minimum value.	Feedback clock delayed @133 MHz and 100 MHz (DQSE=0, DLYBYP=0). The DLL must be used for RX delay. Feedback clock @66 MHz (DQSE=0, DLYBYP=1). The DLL is not used for RX delay.
Micron Xcella (MT35XL01 GBBA ds)	DTR	yes	133	0.8 / 0.8 ns Wrt memory clock edges	2 ns	if kernel clock is a multiple of memory clock, set DHQC = 1 (this automatically bypasses the TX delay). If kernel clock is equal to memory clock, set DQHC = 0. The DLL must be used for TX delay.	DQS delayed (DQSE = 1, DLYBYP = 0). The DLL must be used for RX delay.

DHQC is bit #28 in OCTOSPI_TCR register. DQSE is bit #29 in OCTOSPI_CCR register. DLYBYP is bit #3 in OCTOSPI_DCR1 register.

Enabling the DLL for frequencies greater or equal to 50 MHz (lock mode)

The DLL is enabled by setting EN = 1. The DLL lock state is reached when LOCK = 1. Once the DLL is locked, an RX delay value can be selected by programming RX_TAP_SEL[5:0], and a TX delay value can be selected by programming TX_TAP_SEL[5:0]. Each tap corresponds to 1/32th of the memory device clock period. The selected delays are applied to RX sampling strobe and TX launch clock once RX_TAP_SEL_ACK and TX_TAP_SEL_ACK are set to 1.

Enabling the DLL for frequencies lower than 50 MHz (bypass mode)

The DLL is enabled by setting BYP_EN = 1. The command applied to every DLL delay unit is defined by BYP_CMD[4:0] bits (the value 0 is reserved and must not be used). RX_TAP_SEL[5:0] and TX_TAP_SEL[5:0] define the number of elementary delays inserted on RX sampling clock and TX launch clock. For minimum delay, RX_TAP_SEL[5:0] (respectively TX_TAP_SEL[5:0]) must be set to 0.

EN, RX_TAP_SEL[5:0], TX_TAP_SEL[5:0], BYP_EN, BYP_CMD[4:0] bits are located in SYSCFG_DLYBOSnCR (n=1,2) registers. LOCK, RX_TAP_SEL_ACK, TX_TAP_SEL_ACK bits are located in SYSCFG_DLYBOSnSR (n=1,2) registers.

31.4 SD/*e*MMC interface

The SD/*e*MMC interface offers three SD/*e*MMC physical ports. The accesses to the SDMMC controllers are protected by RISUP components.

31.4.1 SD/eMMC interface main features

- SDMMC1 interface:
  - supports SD cards, eMMC memories, and SDIO devices. Alternate multiplexing is optimized for SD.
  - 1-bit, 4-bit, 8-bit modes.
  - SD: supports SD 3.01 up to UHS-1 modes (SDR104, DDR50, SDR50, DDR25) and all backward compatible modes.
  - eMMC: supports eMMC 5.1 up to HS200 (HS400 not supported)
  - SDIO: supports SDIO 4.0 and all backward compatible modes.
  - SD or eMMC devices are supported as Boot device.
  - DLL to tune the RX sampling clock.
  - SDMMC1 IO segment is in a dedicated V_{DDIO1} supply domain. Both 1V8 and 3V3 devices are supported. 1V2 is not supported.
  - 3V3 to 1V8 voltage transition for UHS-1 modes of SD card can be supported in two different ways:

    . An SDMMC1_VSEL function is mapped on a GPIO and controls an external voltage regulator to switch from 3V3 to 1V8. This voltage regulator supplies the SDMMC1 IO segment.

    . External Level Shifters are used. The enable of these Level Shifters is controlled by a GPIO and their direction is controlled by SDMMC1_CDIR, SDMMC1_D0DIR and SDMMC1_D123DIR alternate functions.
- SDMMC2 interface:
  - supports SD cards, eMMC memories and SDIO devices. Alternate multiplexing is optimized for eMMC.
  - 1-bit, 4-bit, 8-bit modes.
  - SD: supports SD 3.01 up to UHS-1 modes (SDR104, DDR50, SDR50, DDR25) and all backward compatible modes.
  - eMMC: supports eMMC 5.1 up to HS200 (HS400 not supported).
  - SDIO: supports SDIO 4.0 and all backward compatible modes.
  - SD or eMMC devices are supported as Boot device.
  - DLL to tune the RX sampling clock.
  - SDMMC2 IO segment is in a dedicated VDDEMMC supply domain. Both 1V8 and 3V3 devices are supported. 1V2 is not supported.
  - 3V3 to 1V8 voltage transition for UHS-1 modes of SD card can be supported in two different ways:

    . An SDMMC2_VSEL function is mapped on a GPIO and controls an external voltage regulator to switch from 3V3 to 1V8. This voltage regulator supplies the SDMMC2 IO segment.

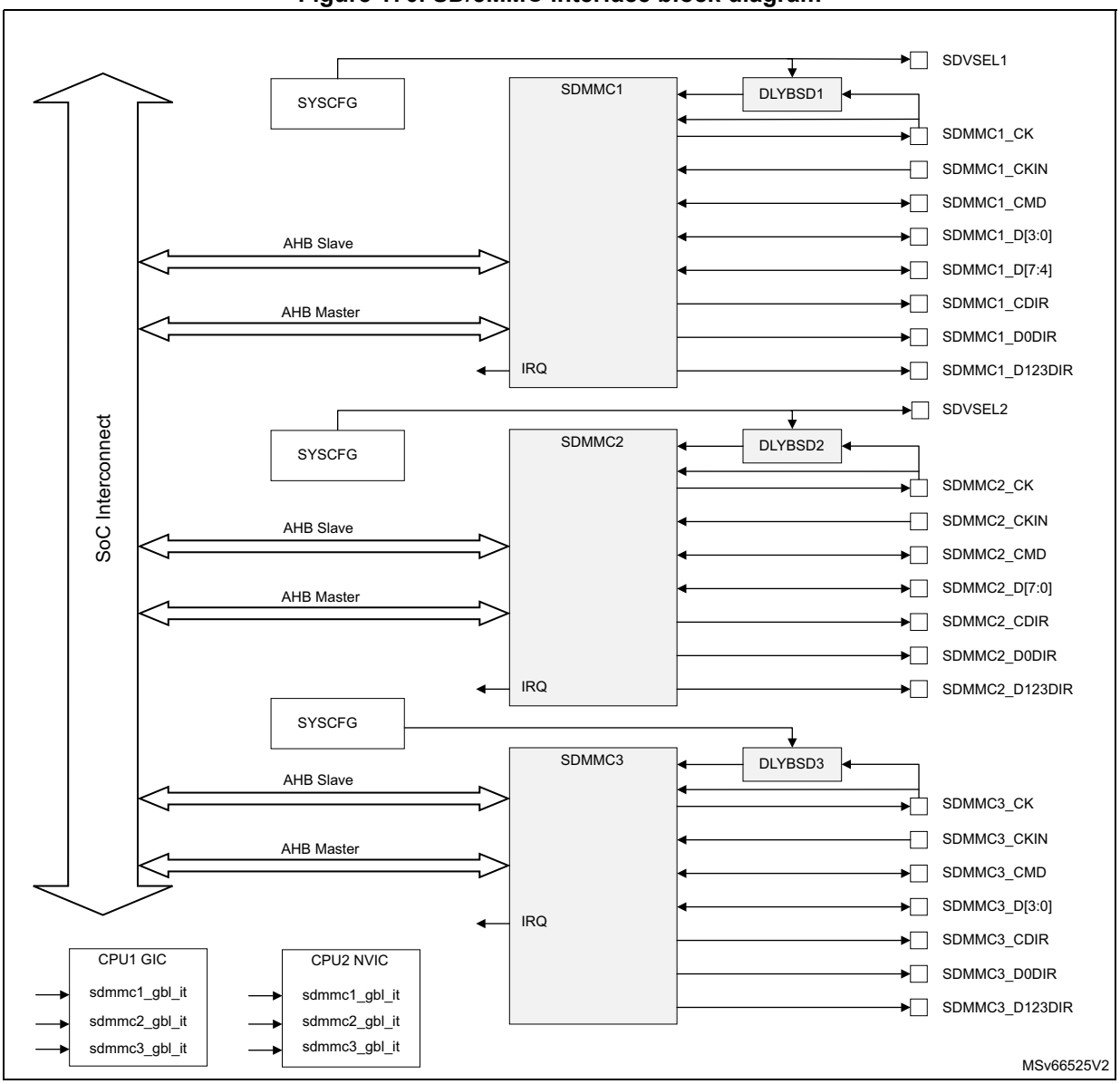
    . External Level Shifters are used. The enable of these Level Shifters is controlled by a GPIO and their direction is controlled by SDMMC2_CDIR, SDMMC2_D0DIR and SDMMC2_D123DIR alternate functions.

- SDMMC3 interface:
  - supports SD cards, eMMC memories (limited to 4-bit) and SDIO devices.
  - 1-bit, 4-bit modes.
  - SD: supports SD 3.01 up to UHS-1 modes (SDR104, DDR50, SDR50, DDR25) and all backward compatible modes.
  - eMMC: supports eMMC 5.1 up to HS200 (HS400 not supported).
  - SDIO: supports up to SDIO 4.0 and all backward compatible modes.
  - DLL to tune the RX sampling clock.
  - SDMMC3 IOs do not have a dedicated supply. They are in VDD IO domain. The external device must also be in VDD IO domain or external Level Shifters must be used.
  - 3V3 to 1V8 voltage transition for UHS-1 modes of SD card can only be supported with external Level Shifters. Like for the other SDMMC interfaces, the enable of these Level Shifters is controlled by a GPIO and their direction is controlled by SDMMC3_CDIR, SDMMC3_D0DIR and SDMMC3_D123DIR alternate functions.

For a complete list of features refer to *Section 34: Secure digital input/output MultiMediaCard interface (SDMMC)*.

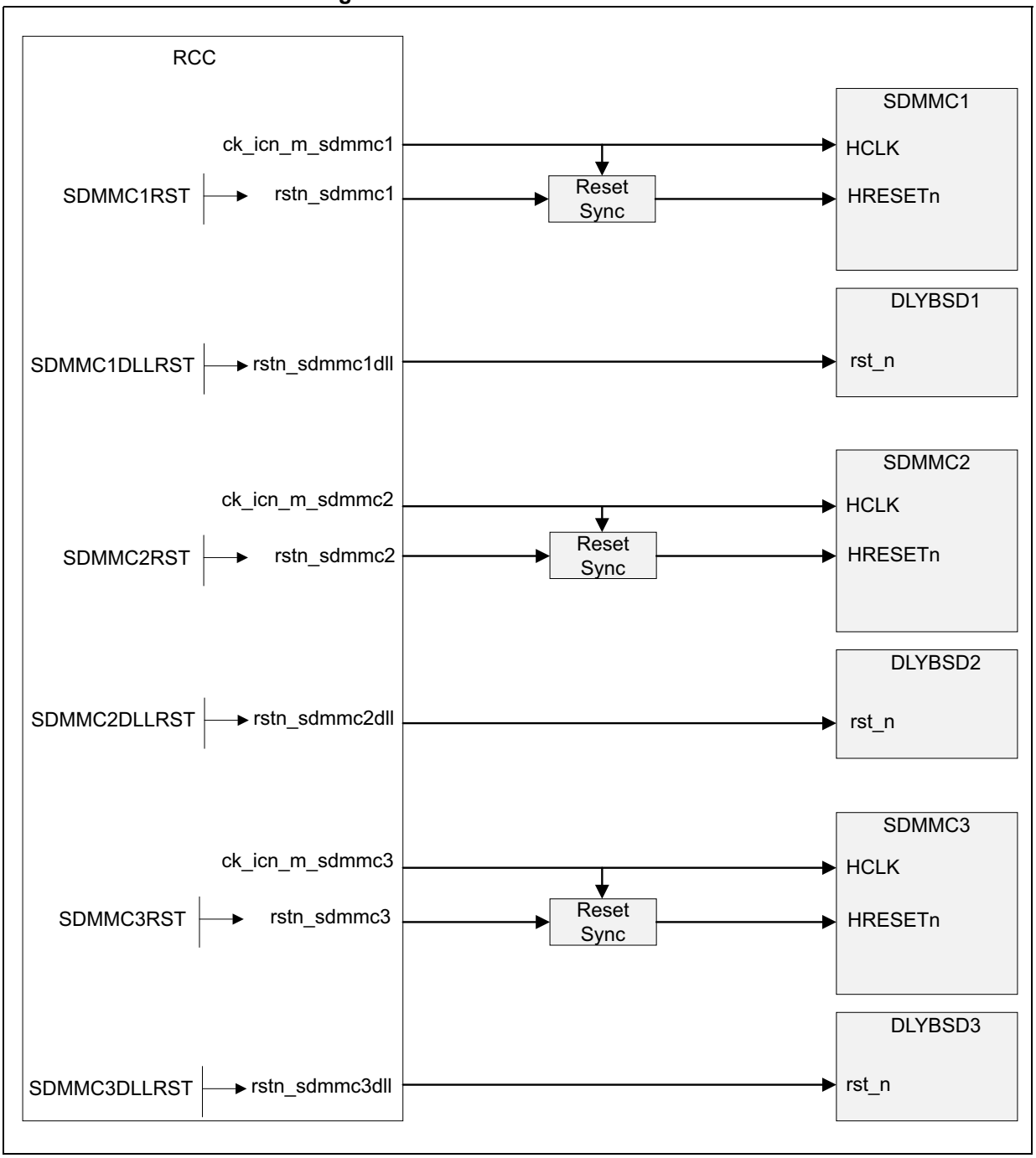
31.4.2 SD/eMMC interface block diagram

Figure 170. SD/eMMC interface block diagram



31.4.3 SD/eMMC interface reset

Figure 171. SD/eMMC interface resets



31.4.4 SDMMC1/2 description

Refer to *Section 34: Secure digital input/output MultiMediaCard interface (SDMMC)*.

31.4.5 DLYBSD1/2 description

One programmable delay is used for each SDMMC interface. The programmable delay can be used to tune the phase of the RX sampling clock in order to sample the data in their valid window. The programmable delay has no effect on the TX path.

These programmable delays are identical to the ones used in the OCTOSPI interfaces. The clock on which the DLL locks is a free running clock provided by the SDMMC controller which is at the same frequency as the clock sent to the memory. When an SDMMC interface operates below 50 MHz, the programmable delay cannot be used. Refer to *Section 31.3.7* for a detailed description.

31.4.6 SD/eMMC interface interrupts

Each SDMMC instance generates one interrupt.

Table 214. SD/eMMC interface interrupts

Peripheral	pin name	CPU1 interrupt	CPU2 interrupt
SDMMC1	IRQ	sdmmc1_gbl_it	sdmmc1_gbl_it
SDMMC2	IRQ	sdmmc2_gbl_it	sdmmc2_gbl_it
SDMMC3	IRQ	sdmmc3_gbl_it	sdmmc3_gbl_it

31.4.7 SD/eMMC interface programmable parameters

Control of external voltage regulator

For SDMMC1 and SDMMC2 interfaces, an external voltage regulator can be used to manage the 3V3 to 1V8 voltage transition for UHS-1 modes of SD card. This external voltage regulator can be controlled by the SDMMC1_VSEL (respectively SDMMC2_VSEL) alternate function. These alternate functions are controlled by the following SYSCFG bits:

- SDMMC1 interface: SDMMC1_VSEL bit in SYSCFG_SDMMC1CR register.
- SDMMC2 interface: SDMMC2_VSEL bit in SYSCFG_SDMMC2CR register.

PWR_CR8.VDDIO1VRSEL (respectively PWR_CR7.VDDIO2VRSEL) must be controlled according to external voltage regulator transition.

DLYBSD1 control

DLYBSD1 programmable delay is controlled by bits in SYSCFG_DLYBSD1CR and SYSCFG_DLYBSD1SR registers. The table below gives the correspondence between the SYSCFG bitfields and the control parameters described in *Section 31.3.7*.

Table 215. DLYBSD1 SYSCFG bits

DLYBSD1 SYSCFG bits	Control input or status output in the SDMMC	Comment
BYP_EN	dll_bypass_cmd_en	Selection of lock or pure-delay mode
EN	dll_start_lock	Enable of DLL when configured in Lock mode
RX_TAP_SEL[5:0]	rx_ph_select[5:0]	Selection of delay in RX delay line

Table 215. DLYBSD1 SYSCFG bits (continued)

DLYBSD1 SYSCFG bits	Control input or status output in the SDMMC	Comment
BYP_CMD[4:0]	dll_bypass_cmd[4:0]	Command applied to every delay units in RX and TX delay lines when the DLL is configured in pure-delay mode
LOCK	dll_locked	When in lock mode, indicates that the DLL has locked on the input clock frequency
RX_TAP_SEL_ACK	rx_ph_select_ack	Indicates that the RX delay line has switched to the delay selected by RX_TAP_SEL[5:0]

DLYBSD2 control

DLYBSD2 programmable delay is controlled by bits in SYSCFG_DLYBSD2CR and SYSCFG_DLYBSD2SR registers. The table below gives the correspondence between the SYSCFG bitfields and the control parameters described in *Section 31.3.7*.

Table 216. DLYBSD2 SYSCFG bits

DLYBSD2 SYSCFG bits	Control input or status output in the SDMMC	Comment
BYP_EN	dll_bypass_cmd_en	Selection of lock or pure-delay mode
EN	dll_start_lock	Enable of DLL when configured in Lock mode
RX_TAP_SEL[5:0]	rx_ph_select[5:0]	Selection of delay in RX delay line
BYP_CMD[4:0]	dll_bypass_cmd[4:0]	Command applied to every delay units in RX and TX delay lines when the DLL is configured in pure-delay mode
LOCK	dll_locked	When in lock mode, indicates that the DLL has locked on the input clock frequency
RX_TAP_SEL_ACK	rx_ph_select_ack	Indicates that the RX delay line has switched to the delay selected by RX_TAP_SEL[5:0]

DLYBSD3 control

DLYBSD3 programmable delay is controlled by bits in SYSCFG_DLYBSD3CR and SYSCFG_DLYBSD3SR registers. The table below gives the correspondence between the SYSCFG bitfields and the control parameters described in *Section 31.3.7*.

Table 217. DLYBSD3 SYSCFG bits

DLYBSD3 SYSCFG bits	Control input or status output in SDMMC	Comment
BYP_EN	dll_bypass_cmd_en	Selection of lock or pure-delay mode
EN	dll_start_lock	Enable of DLL when configured in Lock mode
RX_TAP_SEL[5:0]	rx_ph_select[5:0]	Selection of delay in RX delay line
BYP_CMD[4:0]	dll_bypass_cmd[4:0]	Command applied to every delay units in RX and TX delay lines when the DLL is configured in pure-delay mode
LOCK	dll_locked	When in lock mode, indicates that the DLL has locked on the input clock frequency
RX_TAP_SEL_ACK	rx_ph_select_ack	Indicates that the RX delay line has switched to the delay selected by RX_TAP_SEL[5:0]

31.4.8 Guidelines for selecting the RX sampling clock

The tables below list the various SD/eMMC speed modes and gives guidelines to select the RX sampling clock and configure the clock control bits in the SDMMC cell. It is assumed here that no external level shifters are used. Some information is also given on how to enable the DLL.

Table 218. SD speed modes and recommended clock configuration

Speed Mode	Max card clock freq. SDMMCn_CK	IO Voltage	Setup/hold constraints on SD device	Data valid Window at output of SD Device	Kernel clock freq (SCLK)	DDR	NEGEDGE	CLKDIV	Selection of RX sampling clock (SELCLKRX)
DS (default speed)(1)	25 MHz	3V3	5ns/5ns	26ns	25 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
HS (high speed)(1)	50 MHz	3V3	6ns/2ns	8.5ns	50 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
SDR12(1)	25 MHz	1V8	3ns/0.8ns	27.5ns	25 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
SDR25(1)	50 MHz	1V8	3ns/0.8ns	7.5ns	50 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
DDR50 (2)	50 MHz	1V8	3ns/0.8ns	4.5ns	50 MHz	1	(3)	1 (div by 2)	SDMMC_CLKio
SDR50(2)	100 MHz	1V8	3ns/0.8ns	4ns	100 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
SDR104	208 MHz	1V8	1.4ns/0.8ns	2,88ns	208 MHz	0	NA	0 (div by 1)	SDMMC_FB_CLK

1. For these modes, other clock configurations exist where the kernel clock frequency is a multiple of the card clock. However, considering that all three SDMMC controllers have dedicated kernel clocks, it seems more straightforward to align the kernel clock frequency with the card clock frequency.
2. For these modes, it is also possible to enable the programmable delay and select the delayed loopback clock (SDMMC_FB_CLK) as RX sampling clock if it is needed to adjust the sampling edge.
3. Value to be programmed depend on timings of the interface. For functional simulation, either value works.

Table 219. eMMC Speed modes and recommended clock configuration

Speed Mode	Max card clock freq. SDMMCn_CK	IO Voltage(1)	Setup/Hold constraints on SD device	Data valid Window at output of eMMC Device	Kernel clock freq (SCLK)	DDR	NEGEDGE	CLKDIV	Selection of RX sampling clock (SELCLKRX)
Legacy compatible(2)	26 MHz	3V3, 1V8, 1V2	3ns/3ns	20ns	26 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
High speed SDR(2)	52 MHz	3V3, 1V8, 1V2	3ns/3ns	8ns	52 MHz	0	NA	0 (div by 1)	SDMMC_CLKio
High Speed DDR(3)	52 MHz	3V3, 1V8, 1V2	CMD: 3ns/3ns DATA: 2.5ns/2.5ns	CMD: 8ns DATA: 4.1ns	104 MHz	1	(4)	1 (div by 2)	SDMMC_CLKio
High speed HS200	200 MHz	1V8, 1V2	1.4ns/0.8ns	0.575 UI (2.88ns at 200 MHz)	200 MHz	0	NA	0 (div by 1)	SDMMC_FB_CLK

1. 1V2 is not supported in STM32MP21xx.
2. For these modes, other clock configurations exist where the kernel clock frequency is a multiple of the card clock. However, considering that all three SDMMC controllers have dedicated kernel clocks, it seems more straightforward to align the kernel clock frequency with the card clock frequency.
3. For this mode, it is also possible to enable the programmable delay and select the delayed loopback clock (SDMMC_FB_CLK) as RX sampling clock if it is needed to adjust the sampling edge.
4. Value to be programmed depend on timings of the interface. For functional simulation, either value works.

DDR, NEGEDGE, CLKDIV and SELCLKRX are bitfields in SDMMC_CLKCR register.

The DLL is enabled by writing BYP_EN to 0 and setting EN to 1. The DLL lock state is reached when the status bit LOCK is equal to 1. Once the DLL is locked, a delay value can be selected by programming RX_TAP_SEL[5:0]. The selected delay is applied to the RX sampling clock when the status bit RX_TAP_SEL_ACK is equal to 1. Theses control and status bits are located in the SYSCFG_DLYBSDnCR and SYSCFG_DLYBSDnSR registers (n=1,2,3).

Sequencing between SDMMC and DLL programming when the DLL is used

1. Enable in SDMMC the generation of the free running clock towards the DLL. This is done as soon as PWRCTRL is set to 0b11.
2. Once the memory frequency is increased to high speed (i.e. after the initialization sequence), reset and enable the DLL. The reset of the DLL is controlled by SDMMCnDLLRST (n=1,2,3) bit in the RCC. The enable of the DLL is controlled by EN bit in SYSCFG_DLYBSDnCR (n=1,2,3) register.
3. Once the DLL is locked (as indicated by LOCK = 1 in SYSCFG_DLYBSDnSR register), select the delayed feedback clock as the RX sampling clock by writing SELCLKRX to 0b10.
4. Select the RX delay value by programming RX_TAP_SEL[5:0] in SYSCFG_DLYBSDnCR .
5. Wait for RX_TAP_SEL_ACK bit to be set to 1 in SYSCFG_DLYBSDnSR register before reading data from memory.

At startup a tuning phase is required where steps 4 and 5 are repeated for every RX_TAP_SEL[5:0] values. This enables to determine a range of delays where data are read successfully. The delay value corresponding to the middle of this valid range must be selected for further operation.

32 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes two memory controllers:

- the NOR/PSRAM memory controller
- the NAND memory controller

32.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/One NAND flash memory
  - PSRAM (4 memory subregions)
  - Ferroelectric RAM (FRAM)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent chip-select control for each memory region
- Independent configuration for each memory region
- Write enable and byte lane select outputs for use with PSRAM, SRAM devices
- External asynchronous wait control
- Security and Isolation compliant with the Resource Isolation Framework (RIF)

At startup the FMC pins must be configured by the user application. The FMC input/output pins which are not used by the application can be used for other purposes.

After reset, the FMC is disabled. Once all the used memory controllers are configured, the FMC must be enabled by setting the FMCEN bit.

The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. If some registers have to be re-configured, the FMC must be disabled (FMCEN bit set to 0x0). However, only few bits can be changed on the fly:

- FMCEN bit
- ECCEN and PBEN bits in FMC_PCR register
- FMC_CSQICR and FMC_CSQIER register bits

- FMC command sequencer registers and FMC BCH registers
- WEN bit in the FMC_PCR register

If some parameters have to be modified while the FMC is enabled, the FMC must be first disabled so that no further access is allowed to any memory controller during register modification. Once all needed configurations are updated, enable the FMC as follows:

1. Disable FMC by resetting the FMCEN bit.
2. Wait ISOST[1:0] bits in FMC_SR register to be set to 0b11.
3. Wait until PEF flag is set in the FMC_SR register.
4. Program the registers with new values.
5. Enable the FMC by setting the FMCEN bit.

32.2 FMC implementation

Table 220. FMC features

Modes/features	FMC
NAND flash chip-select outputs	4
NAND flash RNBx inputs	1
NAND/NOR flash with common WAIT pin	No

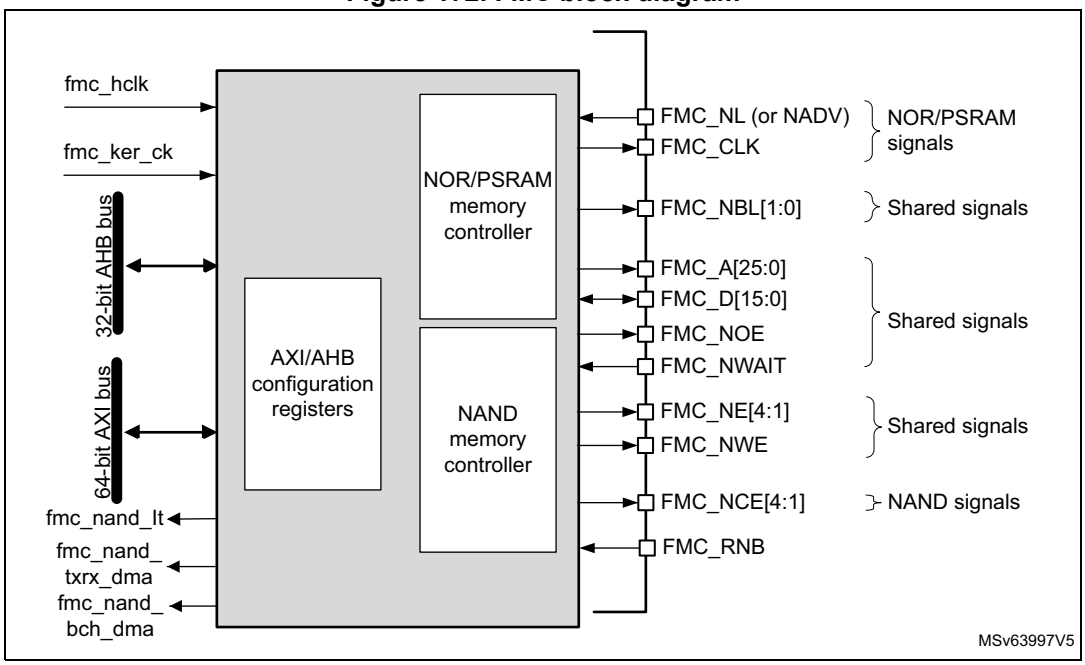
32.3 FMC block diagram

The FMC consists of the following main blocks:

- The NOR flash/PSRAM/SRAM controller
- The NAND controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the below figure.

Figure 172. FMC block diagram



1. The FMC features four NCEx chip select outputs.

32.4 FMC internal signals

Table 221 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 32.10.1: External memory interface signals*.

Table 221. FMC internal signals

Names	Signal type	Description
fmc_nand_it	Digital output	FMC NAND flash interrupt
fmc_ker_ck	Digital input	FMC kernel clock
fmc_hclk	Digital input	FMC interface clock
fmc_nand_txrx_dma	Digital output	DMA request for reading or writing data from/to NAND flash memory
fmc_nand_bch_dma	Digital output	DMA request for BCH results

32.5 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (HCLK) is the reference clock for the FMC register accesses.

The FMC registers must be accessed by words (32 bits). For non 32-bit AHB accesses, FMC does not return an AHB bus error and the register value can be corrupted.

32.6 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32- or 16- or 8-bit accesses according to data size accesses.

The FMC generates an AXI slave error when one of the following conditions is met:

- When reading or writing to an FMC memory region (memory region 1 to 4) which is not enabled
- When reading or writing to any FMC memory region while the FMC is disabled (FMCEN bit is reset)
- When reading or writing by the command sequencer to NAND memory region while it is disabled
- When reading or writing to the NOR flash memory region while the MBKEN bit is reset in the FMC_BCRx register

The kernel clock for the FMC is the asynchronous fmc_ker_ck clock (refer the Reset and Clock control (RCC) section for details on fmc_ker_ck clock source selection).

32.6.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AXI transaction size and memory data size must be equal to prevent issues from occurring.
- AXI transaction size is greater than the memory size:

  In this case, the FMC splits the AXI transaction into smaller consecutive memory accesses to meet the external data width.
- AXI transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM)

    In this case, the FMC allows read/write transactions and accesses the right data through its byte lanes NBL[1:0].

    Bytes to be written are addressed by NBL[1:0].

    All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the flash memory), write transactions are not allowed while read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR flash/PSRAM

Synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read operation is split into two linear burst transactions.
- The write operation is split into two linear burst transactions.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 32.10.6*, for a detailed description of the NOR flash/PSRAM controller registers. Refer to *Section 32.11.9*, for a detailed description of the NAND flash registers.

32.7 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size memory regions of 256 Mbytes each (see *Table 222*):

- Memory region 1 is used to address up to 4 NOR flash memories or PSRAM devices. This region is split into four NOR/PSRAM subregions with four dedicated Chip Selects, as follows:
  - Subregion 1 - NOR/PSRAM 1
  - Subregion 2 - NOR/PSRAM 2
  - Subregion 3- NOR/PSRAM 3
  - Subregion 4 - NOR/PSRAM 4

Note: The Memory region 1 type attribute must be declared as 'device'.

For each memory region, the type of memory to be used can be configured by the user application through the Configuration register.

Table 222. FMC memory regions (default mapping)

Address	Memory region	Supported memory type
0x6000 0000 - 0x6FFF FFFF	Memory region 1	NOR flash memory/PSRAM/SRAM

32.7.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory subregions as shown in *Table 223*.

Table 223. NOR/PSRAM subregion selection

ADDR[27:26](1)	Selected subregion
00	Subregion 1 - NOR/PSRAM 1
01	Subregion 2 - NOR/PSRAM 2
10	Subregion 3- NOR/PSRAM 3
11	Subregion 4- NOR/PSRAM 4

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 224. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	ADDR[25:0]	64 Mbytes * 8 = 512 Mbit
16-bit	ADDR[25:1] >> 1	64 Mbytes/2 * 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC internally uses ADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] must be connected to external memory address A[0].

32.7.2 NAND flash memory address mapping

The 4 Kbyte NAND flash address bits are mapped according to *Table 225*.

Table 225. NAND access memory map

Address field	FMC signal name	Function	Detailed description
ADDR[11:10]	NCEx	NAND chip enable selection	00: NCE1 01: NCE2 10: NCE3 11: NCE4
ADDR[9:8]	-	Reserved	-
ADDR[7]	-	Timing selection	0: Common space, timing defined by FMC_PMEM 1: Attribute space, timing defined by FMC_PATT
ADDR[6]	-	Reserved	-
ADDR[5]	A[17]	Address (ALE)	00: Data section 01: Command section 10: Address section 11: reserved
ADDR[4](1)	A[16]	Command (CLE)	

1. Four address LSBs are don't care: each section has 16 bytes mapped to the alias region.

The application software uses the 3 sections to access the NAND flash memory:

- **To send a command to NAND flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

Note: The NAND region has to be accessed as a device and not as memory. Write accesses to command and address sections must be done in Byte mode and aligned with the section base address (offset = 0).

Read and Write accesses to NAND data section must be in Byte mode for 8-bit NAND flash memories and Byte or Half-word mode for 16-bit NAND flash memories.

Read and Write bursts must start at section base address (offset = 0) and must not cross the 16-byte section boundary.

32.8 FMC security and isolation

The FMC supports security and isolation compliant with the Resource Isolation Framework (RIF). Refer to Section *Resource Isolation Framework* (RIF).

From RIF view point, the FMC is composed of several independent resources, listed below, which can be assigned to different security and compartment domains. They are referenced by an index in the RIF programming model:

- 0: Common FMC_CFGR register
- 1: NOR/PSRAM controller for Chip Select NE1, including registers and associated memory map subregion (64 Mbytes)
- 2: NOR/PSRAM controller for Chip Select NE2, including registers and associated memory map subregion (64 Mbytes)
- 3: NOR/PSRAM controller for Chip Select NE3, including registers and associated memory map subregion (64 Mbytes)
- 4: NOR/PSRAM controller for Chip Select NE4, including registers and associated memory map subregion (64 Mbytes)
- 5: NAND controller, including registers and associated memory map subregion (4 Kbytes)

Note: Only the current NAND controller owner is notified by an interrupt and able to clear it. This owner can be any CID if no filtering is used (CFEN = 0 in FMC_CIDCFGRx), SCID if filtering is used (CFEN = 1) or SEMCID if both filtering and semaphore are used (CFEN = 1 and SEMEN =1) and if SEM_MUTEX = 1. No interrupt is issued when SEM_MUTEX = 0 as there is no owner. In case of sharing with MUTEX, it is mandatory that the owner releases the NAND flash memory resource in a clean state, that is after all pending interrupts have been cleared and masked.

The FMC RIF-compliant security and isolation supports:

- Filtering of secure/non-secure accesses for each resource according to its domain definition.
- Filtering of privileged/unprivileged accesses for each resource according to its domain definition.
- Filtering of compartment identifier (CID) for each resource according to its domain definition
- CID domain definition using dedicated semaphore (optional)
- Locking of security, privilege and CID domain settings for each resource.

Resource access definition is set through dedicated registers:

- FMC_SECCFGR configures the security of local resources.
- FMC_PRIVCFGR configures the privilege of local resources.
- FMC_CIDCFGRx configures compartment access for local resources.
- FMC_SEMCRx is used to configure local semaphore (when used).

By default, all FMC resources are non-secure, unprivileged, they can be accessed in read/write mode by any master, and CID based filtering is disabled.

Trusted domain (TDCID secure, privileged) can configure compartment IDs for FMC resources through FMC_CIDCFRx registers.

RIF-compliant FMC security and Isolation also supports the following features:

- Interrupt gating: a NAND controller interrupt is sent only to the interrupt controller attached to the NAND controller compartment.
- Failed accesses are reported by the illegal access controller (IAC) without bus errors. They are silently ignored. No bus errors are generated by the FMC: failed write operations are ignored, and read operations return zero value on any bus.
- FMC kernel clock gating by RCC is not conditioned by hardware according to FMC resource state. The kernel clock can only be gated after all resources are idle, under application software responsibility.For this purpose local status register bits are read accessible by all domains.

Register accesses are defined in *Table 226*, where:

TD: written only by the trusted domain according to its TDCID

C: bitwise filtering is conditioned to the corresponding register (FMC_SECCFGR,FMC_PRIVCFGR, or FMC_CIDCFGR), all filters must pass.

Y: filtering according to security and privilege

N: no filtering is applied

Table 226. FMC RIF register properties and definitions

Registers	Security	Privilege	CID	Registers used for access definition
FMC_CFGR	C	C	C(1)	FMC_SECCFGR.SEC0, FMC_SECCFGR.PRIV0, FMC_RCFGLOCKR.LOCK0, FMC_CIDCFGR0, FMC_SEMCR0
FMC_BRCR1, FMC_BRTR1, FMC_BWTR1	C	C	C	FMC_SECCFGR.SEC1, FMC_SECCFGR.PRIV1, FMC_RCFGLOCKR.LOCK1, FMC_CIDCFGR1, FMC_SEMCR1
FMC_BRCR2, FMC_BRTR2, FMC_BWTR2	C	C	C	FMC_SECCFGR.SEC2, FMC_SECCFGR.PRIV2, FMC_RCFGLOCKR.LOCK2, FMC_CIDCFGR2, FMC_SEMCR2
FMC_BRCR3, FMC_BRTR3, FMC_BWTR3	C	C	C	FMC_SECCFGR.SEC3, FMC_SECCFGR.PRIV3, FMC_RCFGLOCKR.LOCK3, FMC_CIDCFGR3, FMC_SEMCR3

Table 226. FMC RIF register properties and definitions (continued)

Registers	Security	Privilege	CID	Registers used for access definition
FMC_BRCR4, FMC_BRTR4, FMC_BWTR4	C	C	C	FMC_SECCFGR.SEC4, FMC_SECCFGR.PRIV4, FMC_RCFGLOCKR.LOCK4, FMC_CIDCFGR4, FMC_SEMCR4
FMC_PCR, FMC_PMEM, FMC_PATT, FMC_PATT, FMC_HPR, FMC_HECCR, FMC_CSQCR1/2/3, FMC_CSQAR1/2, FMC_CSQEMSR, FMC_CSQIER, FMC_CSQISR, FMC_CSQICR, FMC_BCHIER, FMC_BCHISR, FMC_BCHICR, FMC_BCHPB1/2/3/4 FMC_BCHDSR0/1/2/3/4	C	C	C	FMC_SECCFGR.SEC5, FMC_SECCFGR.PRIV5, FMC_RCFGLOCKR.LOCK5, FMC_CIDCFGR5, FMC_SEMCR5
FMC_SR, FMC_HWCFGR1, FMC_HWCFGR2, FMC_HWCFGR3, FMC_VERR, FMC_IPIDR, FMC_SIDR	N	N	N	Read -only, read by all
FMC_SECCFGR	Y	Y	C	Bit write only by assigned CID according to FMC_CIDCFGR, FMC_SEMCR and by DDCID according to DDEN. Read by all.
FMC_PRIVCFGR	C	Y	C	Bit write only by assigned CID according to FMC_CIDCFGR, FMC_SEMCR and by DDCID according to DDEN and security configuration. Read by all.
FMC_RCFGLOCKR	Y	Y	TD(1)	Read by all.
FMC_CIDCFGRx	Y	Y	TD(1)	Read by all.
FMC_SEMCRx	C	C	C	Read by all.

1. The debug domain (according to DDCID and RIF DDEN signal) can access all registers when passing secure and privilege filtering.

32.9 FMC RIF registers

32.9.1 FMC security configuration register (FMC_SECCFGR)

Address offset: 0x300

Reset value: 0x0000 0000

Non-secure or unprivileged write accesses to this register are ignored. Read accesses are allowed.

If compartment filtering is enabled (respectively disabled), only trusted domain, secure debug domain (CID) or an allowed secure privileged compartment (respectively any secure privileged compartment) can write to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
										rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SEC5:** Security configuration for NAND controller
0: non-secure
1: secure

Bit 4 **SEC4:** Security state for NOR/PSRAM controller for Chip Select NE4
0: non-secure
1: secure

Bit 3 **SEC3:** Security state for NOR/PSRAM controller for Chip Select NE3
0: non-secure
1: secure

Bit 2 **SEC2:** Security state for NOR/PSRAM controller for Chip Select NE2
0: non-secure
1: secure

Bit 1 **SEC1:** Security state for NOR/PSRAM controller for Chip Select NE1
0: non-secure
1: secure

Bit 0 **SEC0:** Security state for FMC_CFGR common control register
0: non-secure
1: secure

32.9.2 FMC security configuration register (FMC_PRIVCFGR)

Address offset: 0x304

Reset value: 0x0000 0000

Non-secure or unprivileged write accesses to this register are ignored. Read accesses are allowed.

If compartment filtering is enabled (respectively disabled), only trusted domain, secure debug domain (CID) or an allowed secure privileged compartment (respectively any secure privileged compartment) can write to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
										rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **PRIV5:** Privileged state for NAND controller
0: unprivileged
1: privileged

Bit 4 **PRIV4:** Privileged state for NOR/PSRAM controller for Chip Select NE4
0: unprivileged
1: privileged

Bit 3 **PRIV3:** Privileged state for NOR/PSRAM controller for Chip Select NE3
0: unprivileged
1: privileged

Bit 2 **PRIV2:** Privileged state for NOR/PSRAM controller for Chip Select NE2
0: unprivileged
1: privileged

Bit 1 **PRIV1:** Privileged state for NOR/PSRAM controller for Chip Select NE1
0: unprivileged
1: privileged

Bit 0 **PRIV0:** Privileged state for common control: FMC_CFGR
0: unprivileged
1: privileged

32.9.3 FMC CID configuration lock register (FMC_RCFGLOCKR)

Address offset: 0x308

Reset value: 0x0000 0000

Only secure privileged write accesses to this register are allowed. This register can be written only by trusted domain and debug domain (CID). Read accesses are allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
										rs	rs	rs	rs	rs	rs

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **LOCK5:** Lock NAND controller until a global FMC reset
0: no effect
1: FMC_CIDCFGR5, SEC[5] of FMC_SECCFGR and FMC_PRIVCFGR[5] are write locked

Bit 4 **LOCK4:** Lock NOR/PSRAM controller for Chip Select NE4 until a global FMC reset
0: no effect
1: FMC_CIDCFGR4, SEC[4] of FMC_SECCFGR and FMC_PRIVCFGR[4] are write locked

Bit 3 **LOCK3:** Lock NOR/PSRAM controller for Chip Select NE3 until a global FMC reset
0: no effect
1: FMC_CIDCFGR3, SEC[3] of FMC_SECCFGR and FMC_PRIVCFGR[3] are write locked

Bit 2 **LOCK2:** Lock NOR/PSRAM controller for Chip Select NE2 until a global FMC reset
0: no effect
1: FMC_CIDCFGR2, SEC[2] of FMC_SECCFGR and FMC_PRIVCFGR[2] are write locked

Bit 1 **LOCK1:** Lock NOR/PSRAM controller for Chip Select NE1 until a global FMC reset
0: no effect
1: FMC_CIDCFGR1, SEC[1] of FMC_SECCFGR and FMC_PRIVCFGR[1] are write locked

Bit 0 **LOCK0:** Lock FMC_CFGR until a global FMC reset
0: no effect
1: FMC_CIDCFGR0, SEC[0] of FMC_SECCFGR and FMC_PRIVCFGR[0] are write locked

32.9.4 FMC resource x CID register (FMC_CIDCFGRx)

Address offset: 0x30C + 0x8 * x (x = 0 to 5)

Reset value: 0x0000 0000

x designates an FMC resource with associated registers:

x = 0: FMC common control, FMC_CFGR

x = 1: NOR/PSRAM controller for Chip Select NE1 and its memory map address space

x = 2:NOR/PSRAM controller for Chip Select NE2 and its memory map address space

x = 3: NOR/PSRAM controller for Chip Select NE3 and its memory map address space

x = 4: NOR/PSRAM controller for Chip Select NE4 and its memory map address space

x = 5: NAND controller and its memory map address space

Only secure privileged write accesses to this register are allowed. This register can be written only by trusted domain and debug domain (CID) provided LOCKx is cleared in FMC_RCFGLOCKR register.

Read accesses from this register are allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
									rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **SEMWLC7:** White-listed CID in the CID allocation pool for resource x in semaphore mode
0: CID7 is black-listed in CID allocation pool
1: CID7 is white-listed in CID allocation pool

Bit 22 **SEMWLC6:** White-listed CID0 in the CID allocation pool for resource x in semaphore mode
0: CID6 is black-listed in CID allocation pool
1: CID6 is white-listed in CID allocation pool

Bit 21 **SEMWLC5:** White-listed CID0 in the CID allocation pool of resource x in semaphore mode
0: CID5 is black-listed in CID allocation pool
1: CID5 is white-listed in CID allocation pool

Bit 20 **SEMWLC4:** White-listed CID0 in the CID allocation pool of resource x in semaphore mode
0: CID4 is black-listed in CID allocation pool
1: CID4 is white-listed in CID allocation pool

Bit 19 **SEMWLC3:** White-listed CID in the CID allocation pool of resource x in semaphore mode
0: CID3 is black-listed in CID allocation pool
1: CID3 is white-listed in CID allocation pool

Bit 18 **SEMWLC2:** White-listed CID in the CID allocation pool of resource x in semaphore mode
0: CID2 is black-listed in CID allocation pool
1: CID2 is white-listed in CID allocation pool

Bit 17 **SEMWLC1:** White-listed CID0 in the CID allocation pool of resource x in semaphore mode
0: CID1 is black-listed in CID allocation pool
1: CID1 is white-listed in CID allocation pool

Bit 16 **SEMWLC0:** White-listed CID0 in the CID allocation pool of resource x in semaphore mode
0: CID0 is black-listed in CID allocation pool
1: CID0 is white-listed in CID allocation pool

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **SCID[2:0]:** Static CID for resource x
Trusted domain applications use this bit to statically define which compartment has access to the resource x when SEMEN=0.
This bit is ignored if SEMEN=1 or CFEN=0.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEMEN:** Semaphore enable for resource x
0: disable the semaphore mode
1: enable the semaphore mode

Bit 0 **CFEN:** CID filtering enable for resource x
0: disable CID filtering
1: enable CID filtering

32.9.5 FMC resource x semaphore control register (FMC_SEMCRx)

Address offset: 0x310+ x * 0x8 (x = 0 to 5)

Reset value: 0x0000 0000

x designates an FMC resource with associated registers:

x = 0: FMC common control, FMC_SEMCR

x = 1: NOR/PSRAM controller for Chip Select NE1 and its memory map address space

x = 2:NOR/PSRAM controller for Chip Select NE2 and its memory map address space

x = 3: NOR/PSRAM controller for Chip Select NE3 and its memory map address space

x = 4: NOR/PSRAM controller for Chip Select NE4 and its memory map address space

x = 5: NAND controller and its memory map address space

When the secure, privileged and trusted domain has set up resource x in semaphore mode (SEMEN=1 in FMC_CIDCFGRx), the FMC_SEMCRx register can be used during runtime by any authorized white-listed CID agent to take and release the control of resource x.

Read accesses can be privileged or unprivileged, secure or non-secure, and with any CID.

Write accesses are ignored if CFEN = 0 or SEMEN = 0 in FMC_CIDCFGRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
									r	r	r				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SEMCID[2:0]:** Current CID allocated to resource x in semaphore mode

The field is read-only. It is internally updated when a white-listed CID has taken the control of resource x, in semaphore mode. If SEM_MUTEX=0, this CID also released it.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX:** Mutex for the CID allocation of resource x in semaphore mode

Write access:

Write 0: Control of resource x (in semaphore mode) released to any white-listed CID

Write 1: Control of resource x (in semaphore mode) acquired by a white-listed CID

Read access:

Read 0: Resource x is CID-free, meaning that it is not (currently) under the control of any white-listed CID

Read 1: Resource x is CID-allocated, meaning that it is (currently) under the control of one white-listed CID

If resource x is in secure state (SECx=1 in FMC_SECCFGR), this bit can be written only by a secure agent.

If resource x is in privileged state (FMC_PRIVCFGR.PRIVx=1), this bit can be written only by a privileged agent.

If resource x is CID-filtered (CFEN=1 in FMC_CIDCFGRx) and in semaphore mode (FMC_CIDCFGR.SEMEN=1), this bit can be written either by an authorized (white-listed) CID agent to acquire the control or by the same (white-listed) CID agent to release the control.

32.10 NOR flash/PSRAM controller

Caution: The FMC/NOR non-muxed mode is not supported on the STM32MP21xx devices. The muxed mode is enabled by default. The ASYNCWAIT and MUXEN bits of the FMC_BCRx registers, which are not applicable in muxed mode, must be left at their default value.

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per memory region. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait-states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a submultiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, memory region 1 must be configured in synchronous mode (see *Section 32.10.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID[1:0]) otherwise the FMC_CLK frequency is changed depending on AXI data transaction (refer to *Section 32.10.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each memory region is fixed and equal to 64 Mbytes. Each memory region is configured through dedicated registers (see *Section 32.10.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 227*) and support for wait management (for PSRAM and NOR flash accessed in burst mode).

Table 227. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	FMC clock cycle (fmc_ker_ck)	1	15
NBL setup	Duration of the Byte lanes setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Data setup	Duration of the data setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	1	256
Data hold	Duration of the data hold phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	FMC clock cycle (fmc_ker_ck)	0	15
Clock divide ratio	Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)	Synchronous	FMC clock cycle (fmc_ker_ck)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (fmc_ker_ck)	2	17

32.10.1 External memory interface signals

Table 228, *Table 229* and *Table 230* list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals which are active low.

NOR flash memory, non-multiplexed I/Os

Table 228. Non-multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 229. 16-bit multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM/FRAM, non-multiplexed I/Os

Table 230. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 231. 16-bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus

Table 231. 16-bit multiplexed I/O PSRAM (continued)

FMC signal name	I/O	Function
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

32.10.2 Supported memories and transactions

Table 232 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 232. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NOR flash (muxed I/Os and non-muxed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
	Synchronous	R	64	16	Y	Split into 4 FMC accesses

Table 232. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32/64	16	Y	-
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32/64	16	Y	-
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	-
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

32.10.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the fmc_ker_ck clock
- In synchronous mode (read or write), all output signals change on the rising edge of fmc_ker_ck clock.
- NBL signals

  They are applicable to memories which support byte enable.
  - The NBL signals are driven low:

    from the start of a read memory transfer and optionally earlier according to NBLSET, when the device requires some setup time from NBL to NCE,

    for bytes to be written during data transfer and according to the first data bytes during the address transfer.
  - The NBL signals are driven high two FMC_CLK clock cycles after the completion of the read or write transfer.

32.10.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock fmc_ker_ck. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal NE. This guarantees that the memory data hold timing constraint is met (minimum chip enable high to data transition is usually 0 ns).
- If the extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B. The extended mode is not compatible with the muxed mode (EXTMOD bit must be kept cleared in muxed mode).
- If the extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP[1:0] = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP[1:0] = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 173. Mode 1 read access waveforms

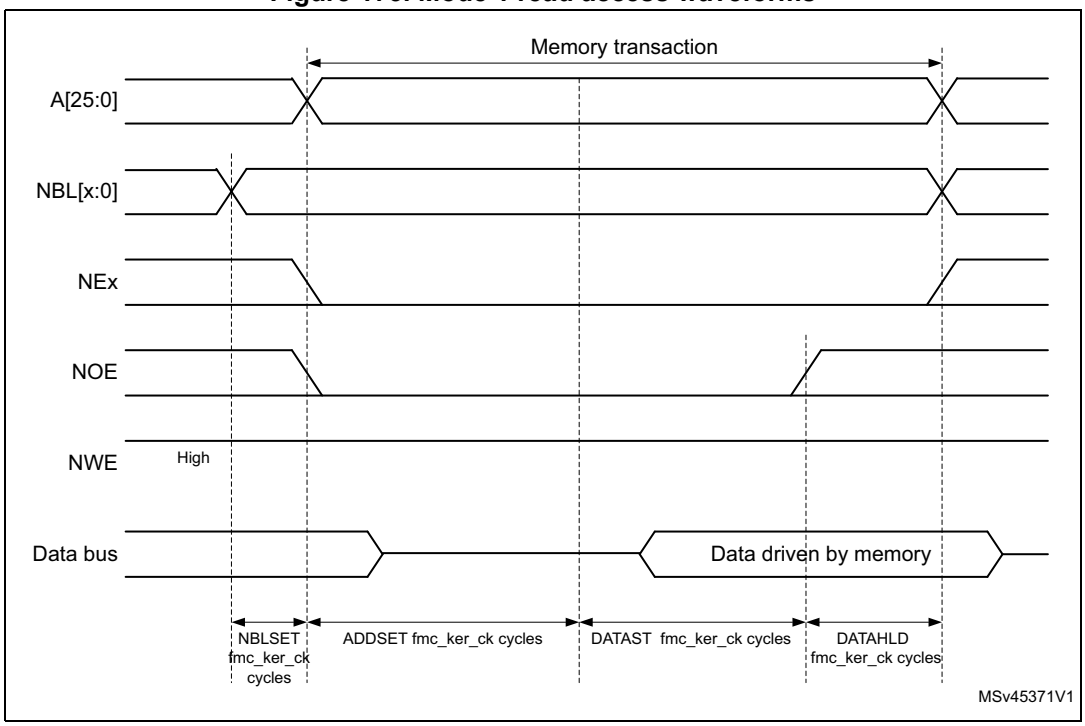
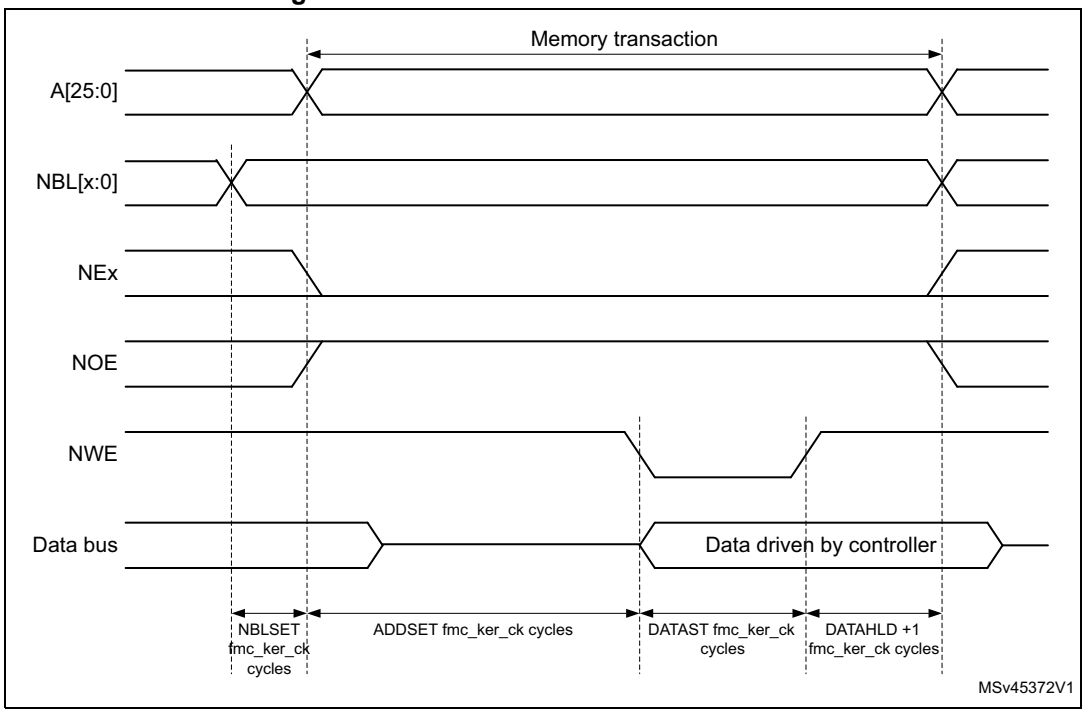


Figure 174. Mode 1 write access waveforms



The DATAHLD time at the end of the read and write transaction guarantee the address and data hold time after the NOE/NWE rising edge. For write, the DATAST value must be greater than zero (DATAST > 0).

Table 233. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 234. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles) for read accesses.
29-28	ACCMOD	Don’t care
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles).

Table 234. FMC_BTRx bitfields (mode 1) (continued)

Bit number	Bit name	Value to set
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 175. Mode A read access waveforms

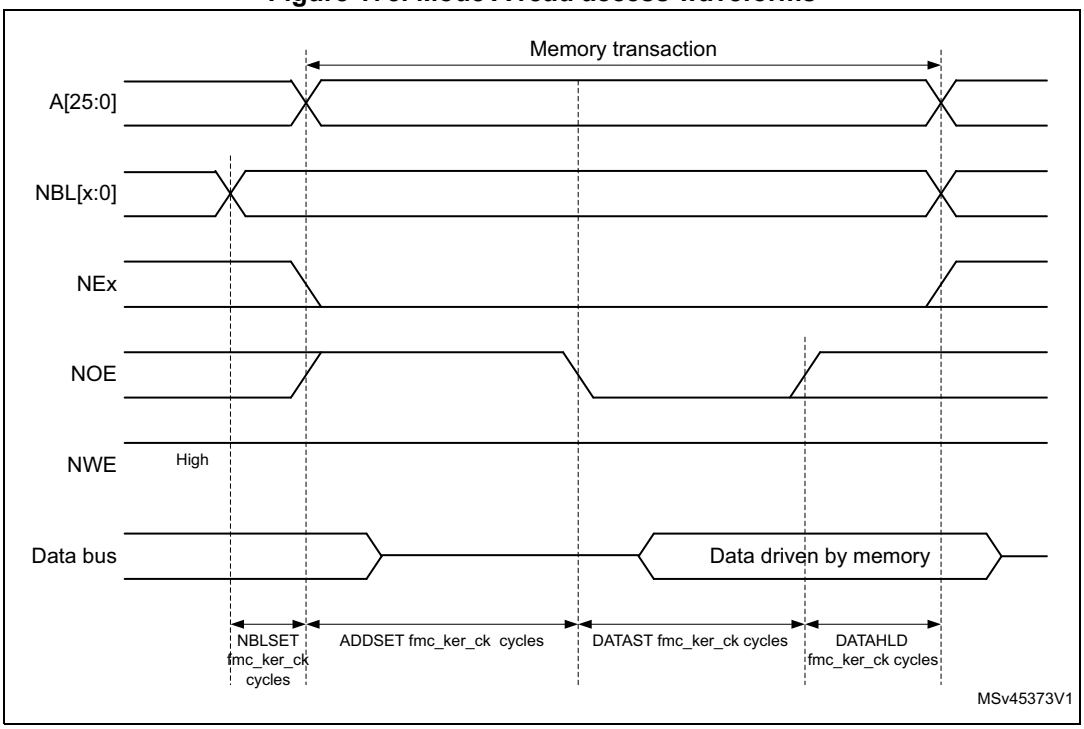
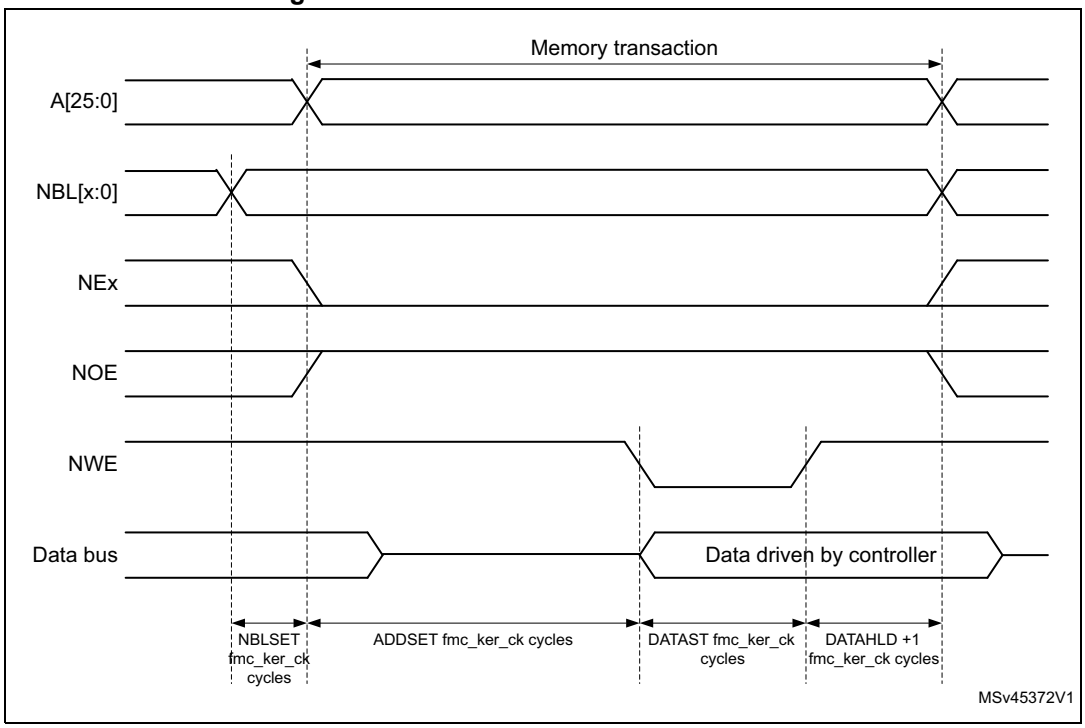


Figure 176. Mode A write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 235. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 235. FMC_BCRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 236. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 237. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR flash

Figure 177. Mode 2 and mode B read access waveforms

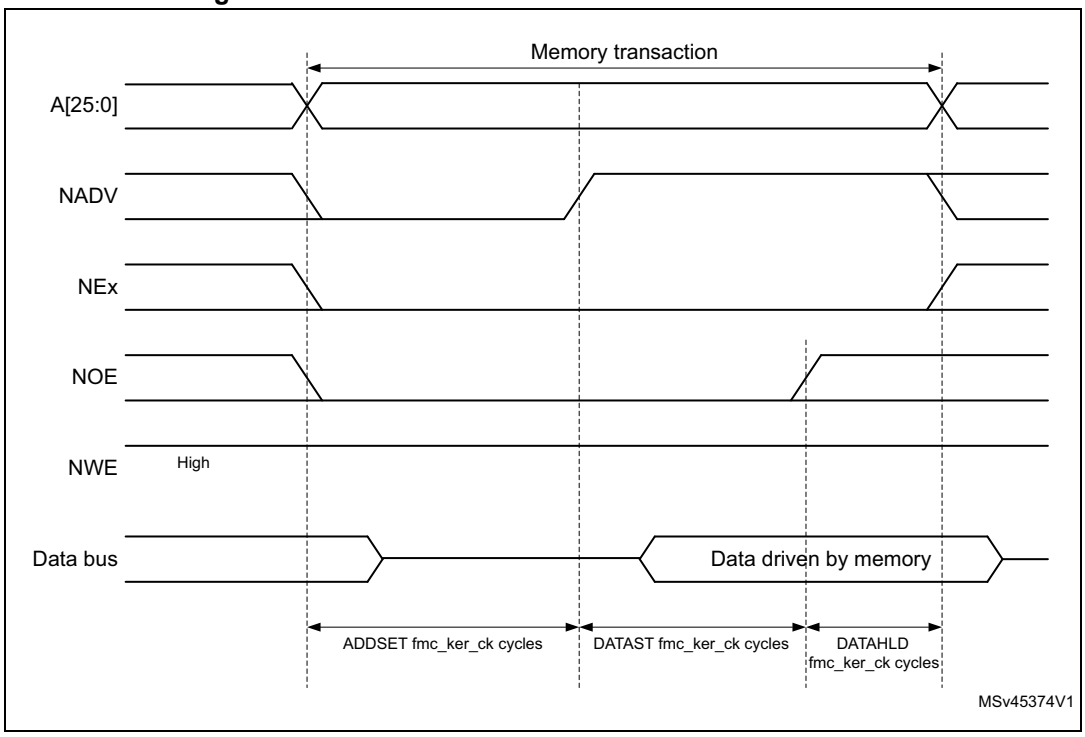


Figure 178. Mode 2 write access waveforms

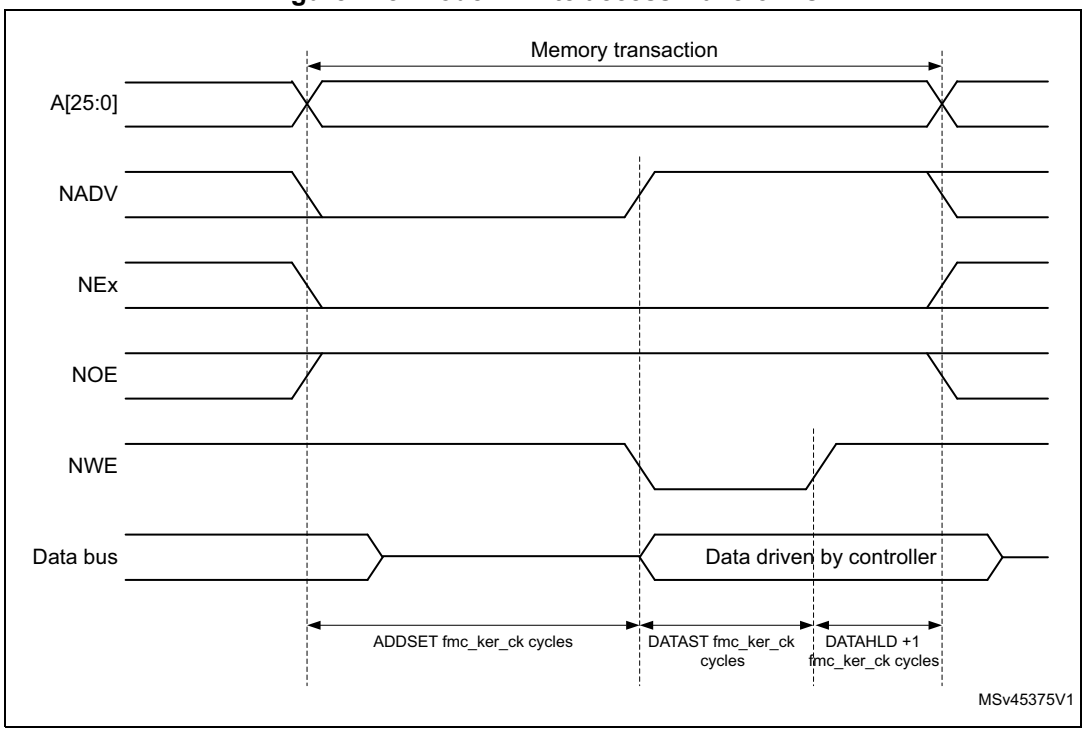
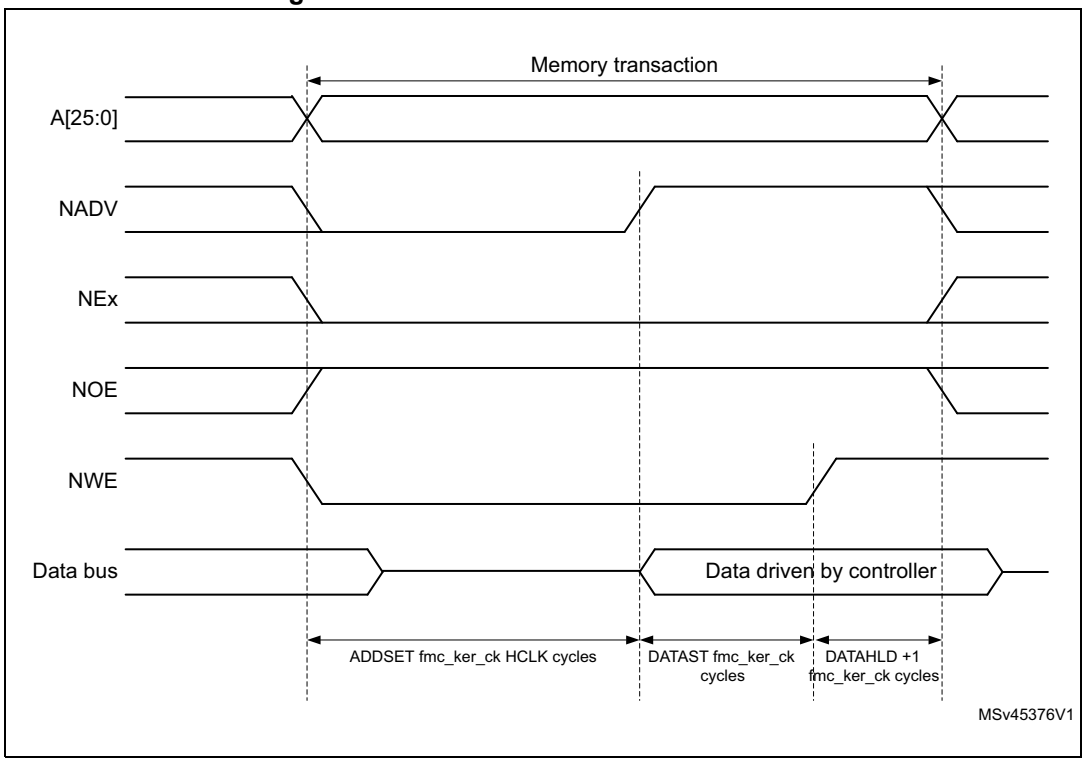


Figure 179. Mode B write access waveforms



The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 238. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 238. FMC_BCRx bitfields (mode 2/B) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 239. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x1 if extended mode is set
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 240. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x1 if extended mode is set
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR flash - OE toggling

Figure 180. Mode C read access waveforms

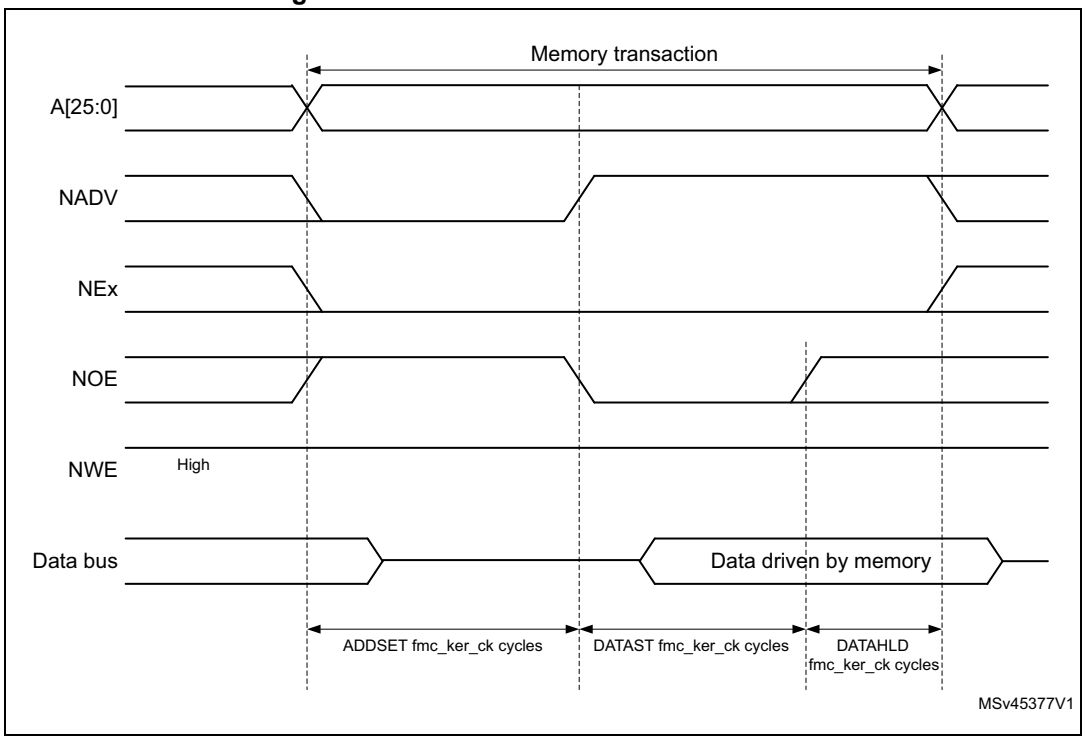
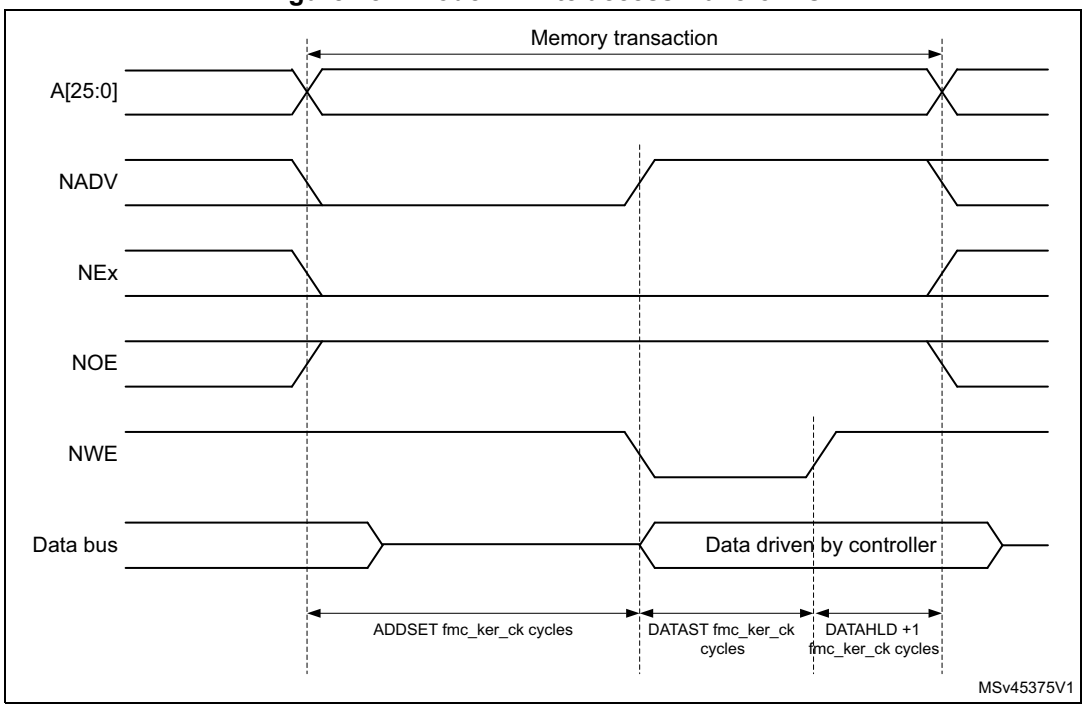


Figure 181. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 241. FMC_BCRx bitfields (mode C)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x02 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 242. FMC_BTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 243. FMC_BWTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x2
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 182. Mode D read access waveforms

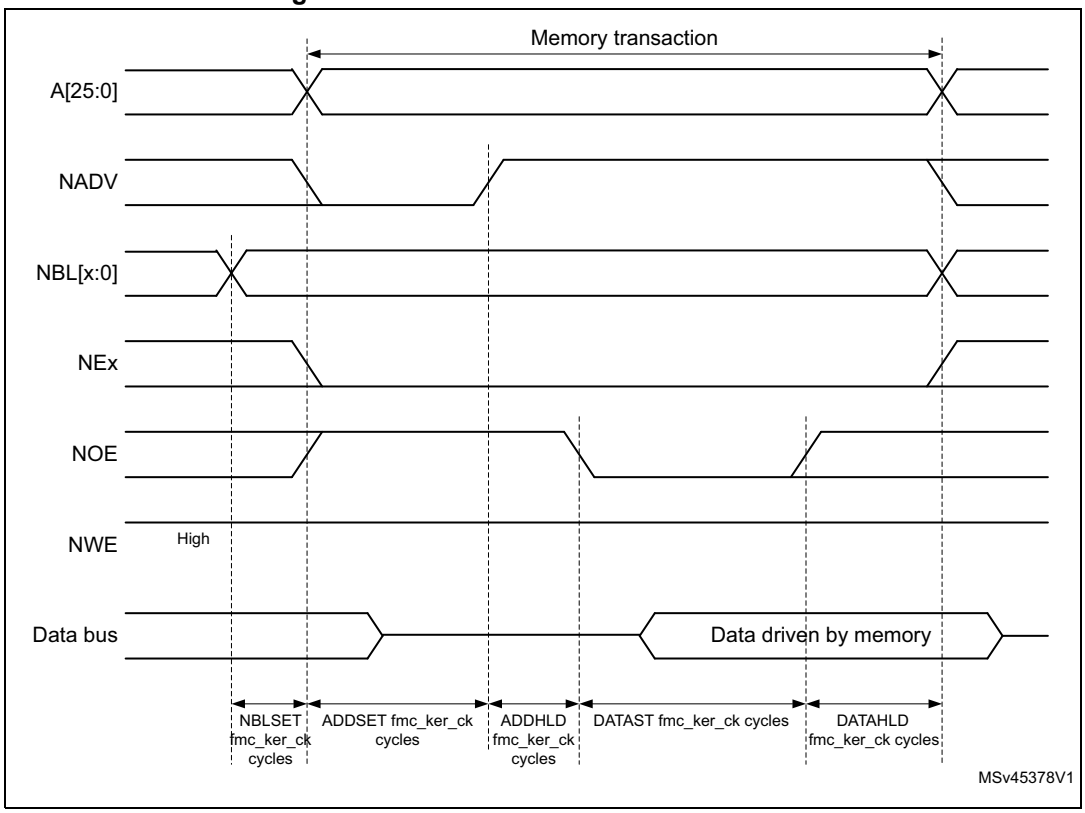
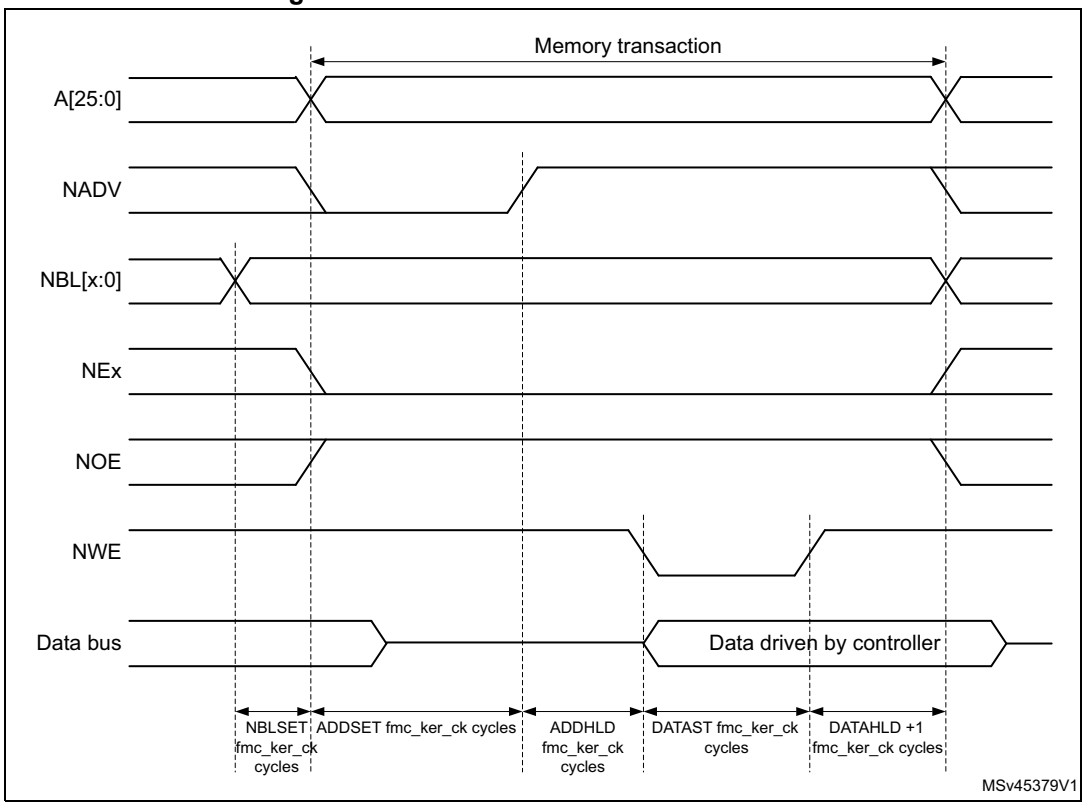


Figure 183. Mode D write access waveforms



The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 244. FMC_BCRx bitfields (mode D)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in mode D mode.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1

Table 244. FMC_BCRx bitfields (mode D) (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 245. FMC_BTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x3
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.

Table 246. FMC_BWTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x3
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 184. Muxed read access waveforms

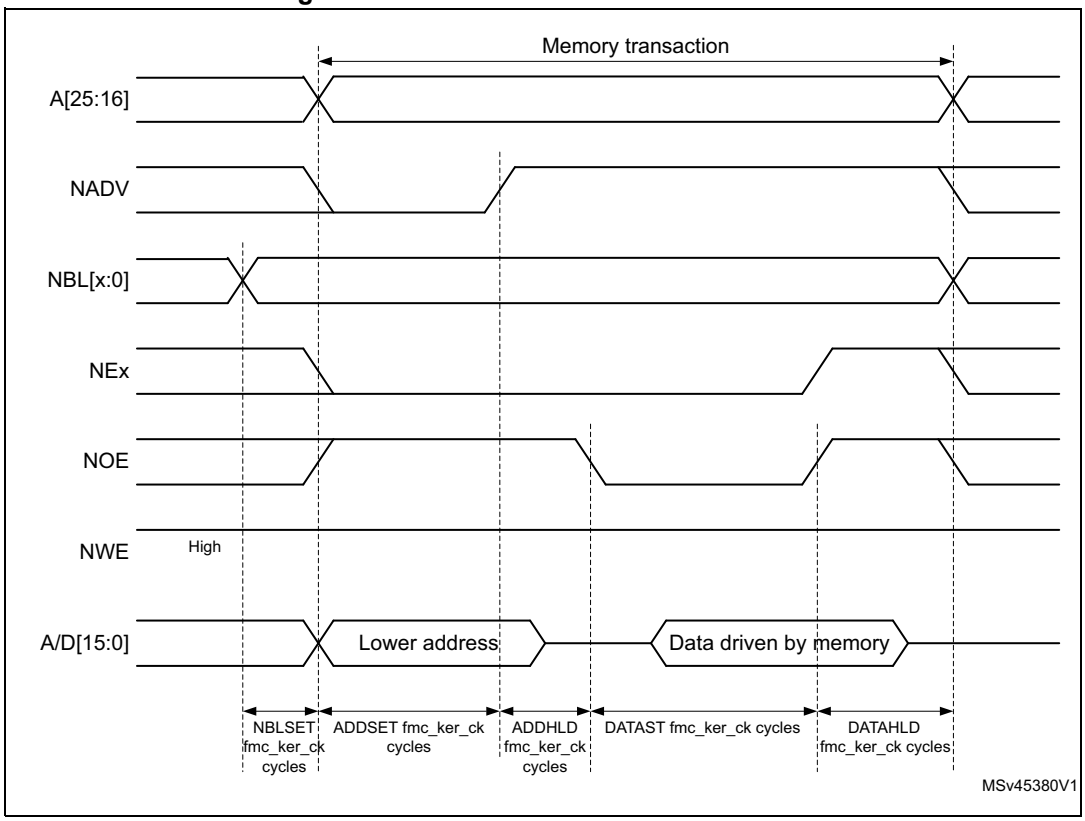
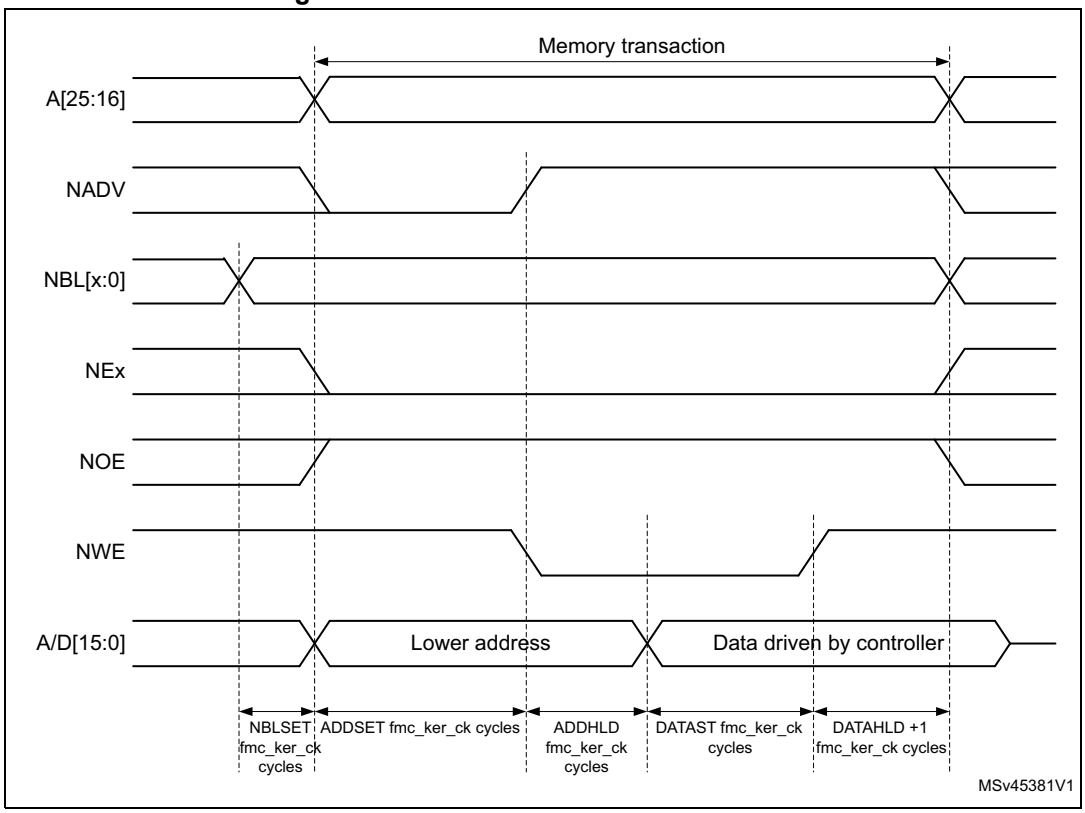


Figure 185. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 247. FMC_BCRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in muxed mode.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1

Table 247. FMC_BCRx bitfields (muxed mode) (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x2 (NOR flash memory)
1	MUXEN	0x1
0	MBKEN	0x1

Table 248. FMC_BTRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses, DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.

WAIT management for asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$DATAST \geq (4 \times FMCCLK) + max_wait_assertion_time$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$max_wait_assertion_time > address_phase + hold_phase$$

   then:

$$DATAST \geq (4 \times FMCCLK) + (max_wait_assertion_time - address_phase - hold_phase$$

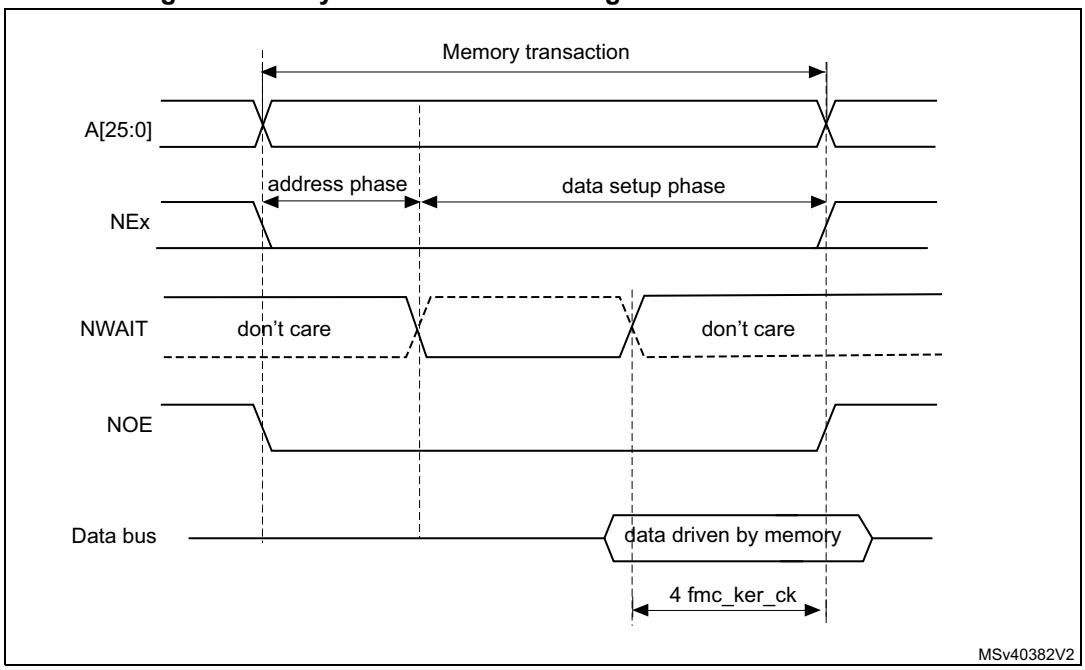
   otherwise

$$DATAST \geq 4 \times FMCCLK$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

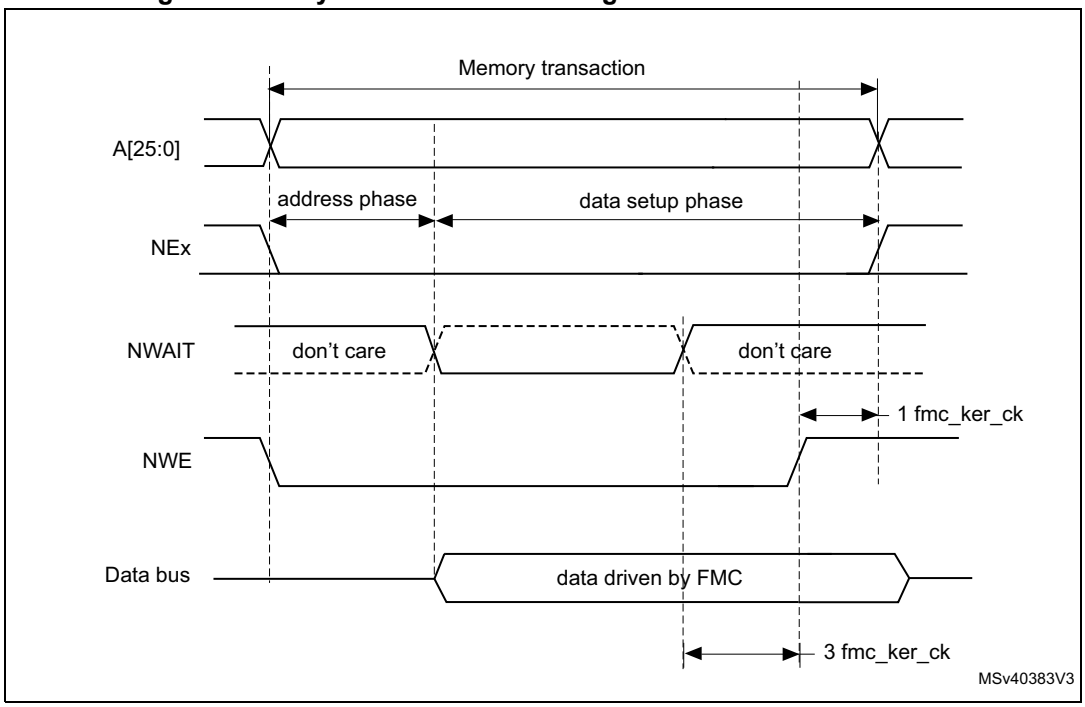
Figure 186 and *Figure 187* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 186. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 187. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not allow maintaining the chip select signal (NE) low for longer than the t_{CEM} timing specified in memory device. This timing can be programmed in the FMC_CFGR register. It defines the maximum duration of the NE low pulse (expressed in fmc_ker_ck cycles) in synchronous mode for PSRAM read or write accesses.

32.10.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of fmc_ker_ck. It depends on the value of CLKDIV.

NOR flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) CLK clock cycles
- or NOR flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected memory region is configured in burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access first requires to re-program the memory access mode, which altogether lasts longer.

Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait-states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait-state (default after reset).
- The flash memory asserts the NWAIT signal during the wait-state

The FMC supports both NOR flash wait-state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0 to 3).

Figure 188. Wait configuration waveforms

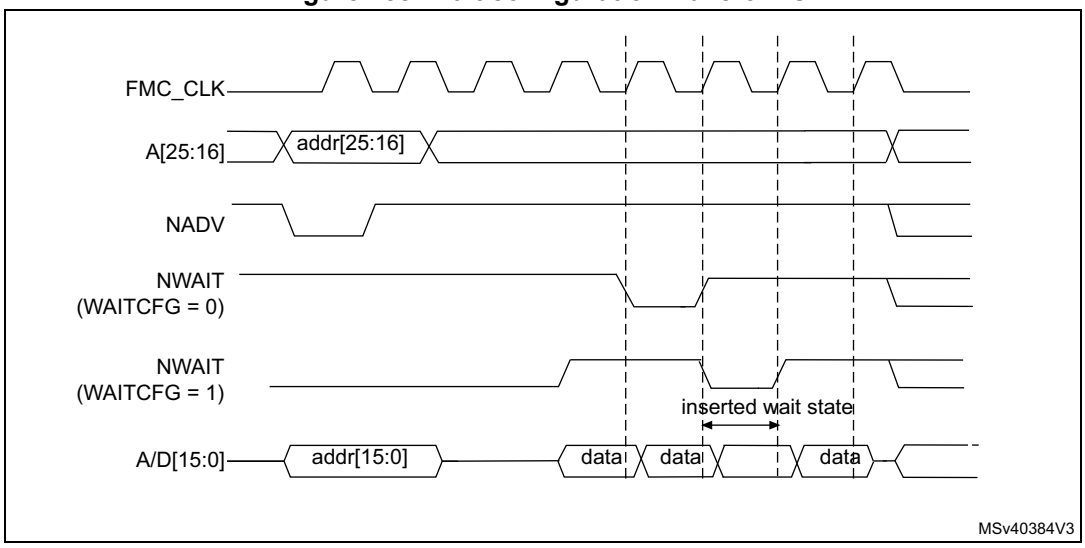


Figure 189. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

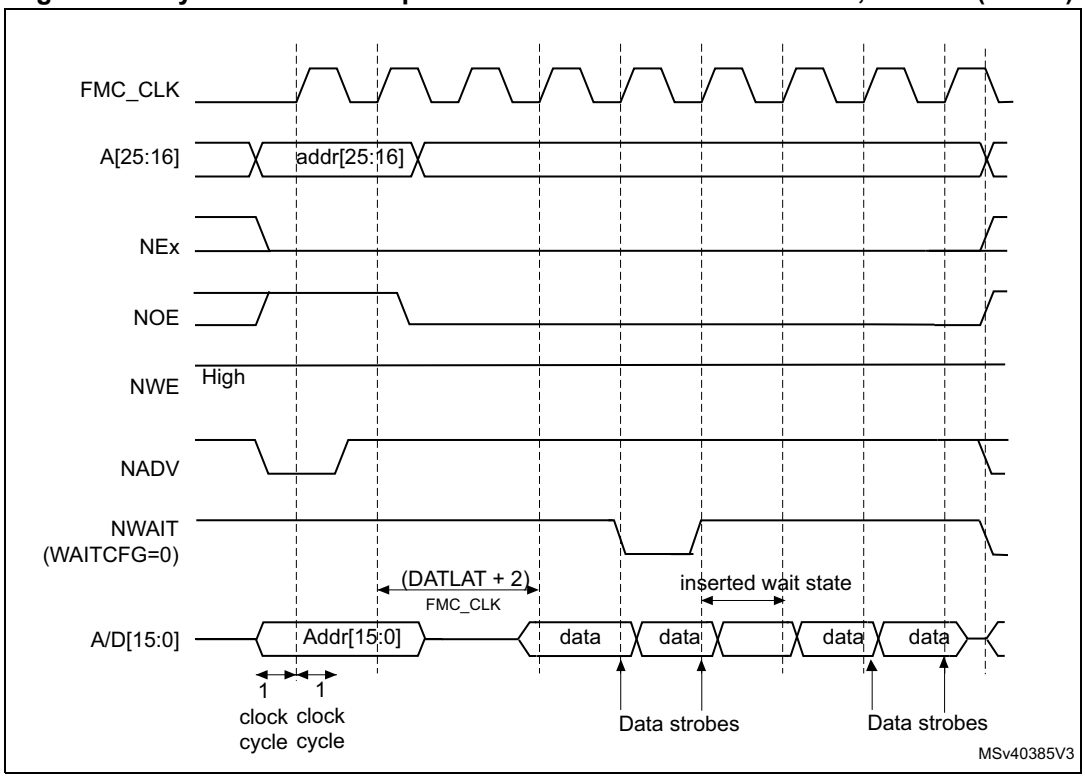


Table 249. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	Reserved	0x0

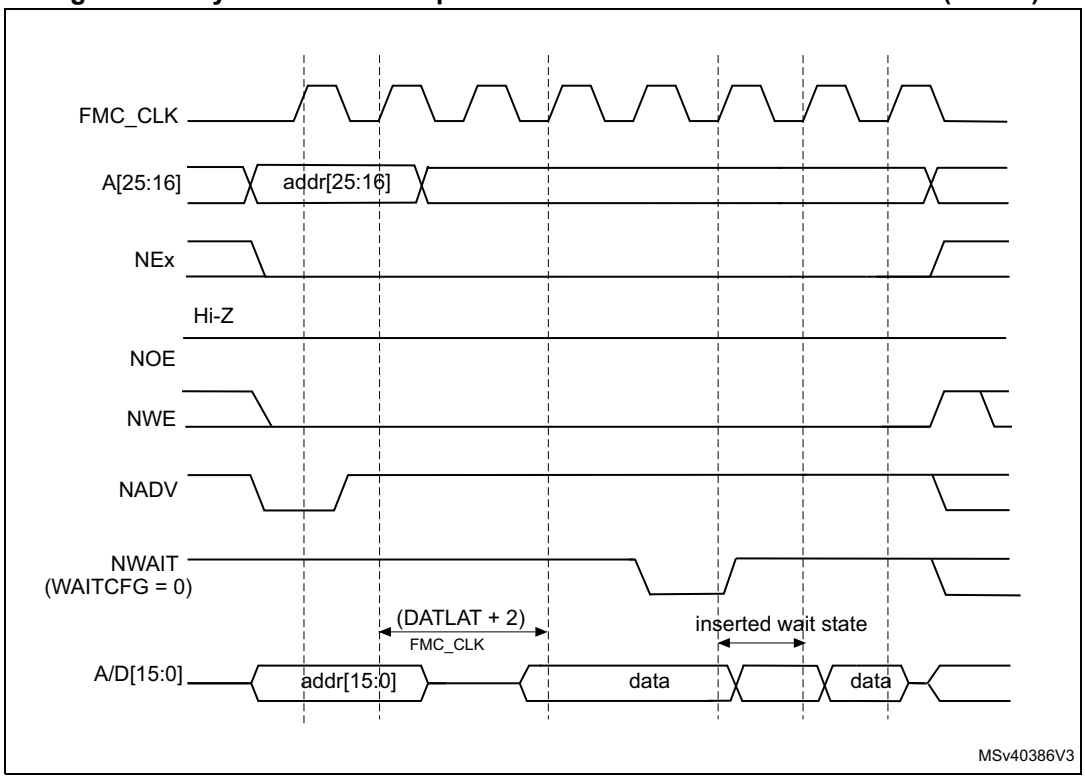
Table 249. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit No.	Bit name	Value to set
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 250. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

Figure 190. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.

Table 251. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1

Table 251. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit No.	Bit name	Value to set
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 252. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don't care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

32.10.6 NOR/PSRAM controller registers

The peripheral registers have to be accessed by words (32-bit)

SRAM/NOR flash chip-select control register for memory region x (FMC_BCRx)

Address offset: 0x00 + 0x08 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory region, used for SRAMs, PSRAM, FRAM and NOR flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
rw	rw	rw	rw	rw		rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **NBLSET[1:0]**: Byte lane (NBL) setup

These bits configure the NBL setup timing from NBLx low to Chip select NEx low.
00: NBL setup time is 0 fmc_ker_ck clock cycle.
01: NBL setup time is 1 fmc_ker_ck clock cycle.
10: NBL setup time is 2 fmc_ker_ck clock cycles.
11: NBL setup time is 3 fmc_ker_ck clock cycles.

Bits 21:20 **CSCOUNT[1:0]:** Chip Select (CS) counter

These bits define the maximum duration of Chip Select low for PSRAM synchronous mode and for each memory region.
The Chip Select counter is loaded into a timer which is decremented using the KCK_FMC while the NE is maintained low during PSRAM read or write access.
When the timer reaches 0, the PSRAM controller splits the current access and toggles NE to allow PSRAM device refresh.
The programmed counter value guarantees a maximum NE pulse width (t_{CEM}) as specified for PSRAM devices.
The counter is reloaded and starts decrementing each time a new access is triggered by a transition of NE from high to low.
0b00: Counter disabled
0b01: NEx deasserted after fmc_ker_ck clock cycle
0b10: NEx deasserted after 64 fmc_ker_ck clock cycles
0b11: NEx deasserted after 256 fmc_ker_ck clock cycles

Note: Most PSRAMs manage refresh transparently and do not require access split. CSCOUNT[1:0] must consequently be left at its default value (counter disabled), otherwise CSCOUNT[1:0] value and access split impact must be checked according to PSRAM timing parameters.

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.
0: Write operations are always performed in asynchronous mode
1: Write operations are performed in synchronous mode.

Bits 18:16 **CPSIZE[2:0]:** CRAM page size

These bits are used for CellularRAM™ 1.5 which does not allow burst accesses to cross the address boundaries between pages. When these bits are configured, the FMC automatically splits the burst access when the memory page size is reached (refer to memory datasheet for page size).
000: No burst split when crossing page boundary. (default after reset).
001: 128 bytes
010: 256 bytes
011: 512 bytes
100: 1024 bytes
Others: Reserved.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables wait signal usage by the FMC even during asynchronous transfers.
0: NWAIT signal not taken in to account during asynchronous transfers (default after reset)
1: NWAIT signal taken in to account during asynchronous transfers

Bit 14 **EXTMOD:** Extended mode enable

This bit enables write timing programming for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: Values inside FMC_BWTR register not taken into account (default after reset)

1: Values inside FMC_BWTR register taken into account

Note: When the extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP[1:0] =0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP[1:0] = 0x10).*

Bit 13 **WAITEN:** Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in synchronous mode.

0: NWAIT signal disabled (its level not taken into account, no wait-state inserted after the programmed flash latency period)

1: NWAIT signal enabled (its level is taken into account after the programmed latency period to insert wait-states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit

This bit indicates whether write operations to the memory region by the FMC are enabled/disabled:

0: Write operations to the memory region by the FMC disabled. An AXI slave error is reported,

1: Write operations to the memory region by the FMC enabled (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait-state must be inserted when accessing the memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait-state or during the wait-state:

0: NWAIT signal active one data cycle before wait-state (default after reset),

1: NWAIT signal active during wait-state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL:** Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in synchronous or asynchronous mode:

0: NWAIT active low (default after reset),

1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in asynchronous mode.

1: Burst mode enable. Read accesses are performed in synchronous mode.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MWID[1:0]:** Memory data bus width
This bit defines the external memory device width, valid for all type of memories.
00: 8 bits
01: 16 bits (default after reset)
10: Reserved.
11: Reserved.

Bits 3:2 **MTYP[1:0]:** Memory type
This bit defines the type of external memory attached to the corresponding memory region:
00: SRAM/ FRAM (default after reset for memory region 2...4)
01: PSRAM (CRAM)/FRAM
10: NOR flash/OneNAND flash (default after reset for memory region 1)
11: Reserved.

Bit 1 **MUXEN:** Address/data multiplexing enable bit
When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/Data non-multiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory region enable bit
This bit enables the memory region. After reset memory region 1 is enabled, all others are disabled. Accessing a disabled memory region causes an ERROR on AXI bus.
0: Corresponding memory region is disabled
1: Corresponding memory region is enabled

SRAM/NOR flash chip-select timing registers for memory region x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory region, used for SRAMs, PSRAM and NOR flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, two registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 186* to *Figure 187*), used in asynchronous accesses:
For read/write accesses:
00: DATAHLD phase duration = fmc_ker_ck clock cycle * 0 (default, read)/1 (default, write)
01: DATAHLD phase duration = fmc_ker_ck clock cycle * 1 (read)/2 (write)
10: DATAHLD phase duration = fmc_ker_ck clock cycle * 2 (read)/3 (write)
11: DATAHLD phase duration = fmc_ker_ck clock cycle * 3 (read)/4 (write)

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
00: Access mode A
01: Access mode B
10: Access mode C
11: Access mode D

Bits 27:24 **DATLAT[3:0]:** Data latency for synchronous memory (see note below bit descriptions)

For synchronous access with read/write burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data.
This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.
For asynchronous access, this value is don't care.
0000: Data latency of 2 FMC_CLK clock cycles for first burst access
1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define FMC_CLK clock output signal period, expressed in fmc_ker_ck cycles:
0000: Reserved.
0001: FMC_CLK period = 2 × fmc_ker_ck periods
0010: FMC_CLK period = 3 × fmc_ker_ck periods
1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)
This value is don't care when accessing NOR flash memories, SRAM and PSRAM in asynchronous mode.

Note: Refer to Section 32.10.5: Synchronous transactions for FMC_CLK *divider ratio formula).*

Note: CLKDIV[3:0] bits are don't care when CCLKEN bit of FMC_CFGR is set. In this case FMC_CLK clock output is controlled by CLKDIV of FMC_CFGR.

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration

These bits are written by software to add a delay between the end of current read or write transaction and the next transaction on the same memory region.

This delay allows matching the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time required by the memory to free the data bus after a read access ($t_{EHQZ\ f}$from chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to a minimum value that respects the following condition:

(BUSTRUN + 1)fmc_ker_ck period ≥ $t_{EHELmin}$ max ($t_{EHELmin}$, $t_{EHQZmax)}$

For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). This delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read) to match the t_{PC} memory timing, and the Chip select toggles between consecutive accesses.

(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}

0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 186* to *Figure 187*) used in asynchronous accesses:
0000 0000: Reserved.
0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles
0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles
...
1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)
For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 186* to *Figure 187*).
Example of mode 1, write access, DATAST=1:
Data-phase duration= DATAST+1 = 2 *fmc_ker_ck clock cycles.

Note: This value is don't care in synchronous accesses.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 182* to *Figure 185*) used in mode D or multiplexed accesses:
0000: Reserved.
0001: ADDHLD phase duration =1 * fmc_ker_ck clock cycle
0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle
...
1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)
For each access mode address-hold phase duration, refer to the respective figure (*Figure 182* to *Figure 185*).

Note: This value is not used in synchronous accesses, and the address hold phase lasts always 1 memory clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 186* to *Figure 187*), used in SRAMs, ROMs and asynchronous NOR flash memories:
0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle
...
1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)
For each access mode address setup phase duration, refer to the respective figure (refer to *Figure 186* to *Figure 187*).

Note: This value is don't care in synchronous accesses.
In muxed mode or mode D, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.

On PSRAMs (CRAMs), DATLAT must be configured to 0 so that the FMC quickly exits its latency phase, starts sampling NWAIT from memory, and starts read or write operations when the memory is ready.

This method can be used also with the latest generation of synchronous flash memories, which, unlike older flash memories, issue the NWAIT signal (check the corresponding flash memory datasheet).

SRAM/NOR-flash write timing registers for memory region x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x000F FFFF

This register contains the control information of each memory region. It is used for SRAMs, FRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]			
rw	rw	rw	rw									rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 187*), used in asynchronous write accesses:

00: DATAHLD phase duration = 1 * fmc_ker_ck clock cycle (default)

01: DATAHLD phase duration = 2 * fmc_ker_ck clock cycle

10: DATAHLD phase duration = 3 * fmc_ker_ck clock cycle

11: DATAHLD phase duration = 4 * fmc_ker_ck clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A

01: Access mode B

10: Access mode C

11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay between the end of current write transaction and the next transaction on the same memory region.

For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). The bus turnaround delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read), and the Chip select toggles between consecutive accesses.

(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}

0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the data phase duration (refer to *Figure 186* to *Figure 187*) used in asynchronous SRAM, PSRAM and NOR flash memory accesses:
0000 0000: Reserved.
0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles
0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles
...
1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the address hold phase (refer to *Figure 184* to *Figure 185*) used in asynchronous multiplexed accesses:
0000: Reserved.
0001: ADDHLD phase duration = 1 * fmc_ker_ck clock cycle
0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle
...
1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, and the address hold phase always lasts 1 flash clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the address setup phase in fmc_ker_ck cycles (refer to *Figure 186* to *Figure 187*) used in asynchronous accesses:
0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle
...
1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, and the address setup phase always lasts 1 flash clock period. In muxed mode, the minimum ADDSET value is 1.

FMC common configuration register (FMC_CFGR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is user to set configuration common to all controllers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCLKEN	CLKDIV[3:0]			
rw											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **FMCEN:** FMC enable
This bit enables/disables the FMC.
0: FMC disabled
1: FMC enabled

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CCLKEN:** Continuous clock enable
This bit enables the FMC_CLK clock output to external memory devices.
0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is defined by the programmed CLKDIV value in the FMC_BCRx register (default after reset).
1: The FMC_CLK is generated continuously during asynchronous and synchronous access.
If continuous clock mode is enabled, the synchronous memories connected to all memory regions are clocked by the same clock with the clock ratio defined by CLKDIV in FMC_CFGR. The CLKDIV value in FMC_BWTRx are don't care.

Bits 19:16 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)
These bits define the FMC_CLK clock output signal period, expressed in number of fmc_ker_ck cycles:
0000: Reserved.
0001: FMC_CLK period = 2 × fmc_ker_ck periods
0010: FMC_CLK period = 3 × fmc_ker_ck periods
1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)
This value is don't care when accessing NOR flash memories, SRAM and PSRAM in asynchronous mode.

Note: Refer to Section 32.10.5: Synchronous transactions for FMC_CLK divider ratio formula.

Bits 15:0 Reserved, must be kept at reset value.

FMC size identification register (FMC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size Identification
These bits return the size of the memory region allocated to the FMC registers.

FMC identification register (FMC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0014 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** FMC Identifier
These bits return the FMC identifier

FMC version register (FMC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision
These bits return the FMC major revision

Bits 3:0 **MINREV[3:0]:** Minor revision
These bits return the FMC minor revision

FMC hardware configuration register 1 (FMC_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x2232 9011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA_LN2DPTH[3:0]				WR_LN2DPTH[3:0]				WD_LN2DPTH[3:0]				WA_LN2DPTH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_SIZE[3:0]				Res.	Res.	Res.	SDRAM_SEL	Res.	Res.	Res.	NAND_ECC	Res.	Res.	Res.	NAND_SEL
r	r	r	r				r				r				r

Bits 31:28 **RA_LN2DPTH[3:0]:** AXI read address FIFO depth
These bits return the AXI read address FIFO depth. depth is $2^{RA_LN2DPTH}$.

Bits 27:24 **WR_LN2DPTH[3:0]:** AXI write response FIFO depth
These bits return the AXI write response FIFO depth. depth is $2^{WR_LN2DPTH}$.

Bits 23:20 **WD_LN2DPTH[3:0]:** AXI write data FIFO depth
These bits return the AXI write data FIFO depth. depth is $2^{WD_LN2DPTH}$.

Bits 19:16 **WA_LN2DPTH[3:0]:** AXI write address FIFO depth
These bits return the AXI write address FIFO depth. depth is $2^{WA_LN2DPTH}$.

Bits 15:12 **ID_SIZE[3:0]:** AXI ID width
These bits return the AXI ID width.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDRAM_SEL:** SDRAM controller selection
0x0: SDRAM controller not supported
0x1: SDRAM controller supported

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **NAND_ECC:** NAND ECC
0x0: Hamming code (supported only in Legacy mode without sequencer)
0x1: Hamming and BCH code

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **NAND_SEL:** NAND Controller Selection
0x0: NAND Controller not supported
0x1: NAND Controller supported

FMC hardware configuration register 2 (FMC_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x11DC 4772

The base addresses are relevant only when the FMC includes the corresponding memory controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SECURE	Res.	Res.	Res.	PRIVILEGE	SDRAM2_BASE[3:0]				SDRAM1_BASE[3:0]			
			r				r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAND_BASE[3:0]				SDRAM_RBASE[3:0]				NOR_BASE[3:0]				RD_LN2DPTH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SECURE**: Support for Arm® TrustZone®

0: no TrustZone

1: Support for TrustZone

Note: Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **PRIVILEGE**: Support for Privileged mode

0: Privileged mode not supported

1: Privileged mode supported

Bits 23:20 **SDRAM2_BASE[3:0]:** SDRAM bank 2 base address

It returns bits [31:28] of SDRAM bank 2 base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 19:16 **SDRAM1_BASE[3:0]:** SDRAM bank 1 base address

It returns bits [31:28] of SDRAM bank 1 base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 15:12 **NAND_BASE[3:0]:** NAND base address

It returns bits [31:28] of NAND base address space

Bits 11:8 **SDRAM_RBASE[3:0]:** SDRAM remap base address

it returns bits [31:28] of SDRAM remap base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 7:4 **NOR_BASE[3:0]:** NOR base address

It returns bits [31:28] of NOR base address space

Bits 3:0 **RD_LN2DPTH[3:0]:** AXI read data FIFO depth

These bits return the AXI read data FIFO depth. depth is $2^{RD_LN2DPTH}$

FMC hardware configuration register 3 (FMC_HWCFGR3)

Address offset: 0x3E8

Reset value: 0x0000 0037

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID_WIDTH[3:0]				MAX_CID[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CID_WIDTH[3:0]**: CID bus width
This bitfield is relevant only when MAX_CID is not null in FMC_HWCFGR2.
0000: Width = 1
0001: Width = 2
0010: Width = 3
0011: Width = 4
Others: Reserved, must not be used.

Bits 3:0 **MAX_CID[3:0]**: Maximum compartment ID (or CID)
MAX_CID ranges from 0 to 15:
0000: no CID supported, DMA is not CID-aware
0001 to 1111: FMC is CID-aware and each channel is a separate CID-aware resource. n+1 CIDs, from CID0 up to CIDn, are supported.

32.11 NAND flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND flash memories

The NAND memory region is configured through dedicated registers (*Section 32.11.9*). The programmable memory parameters include access timings (shown in *Table 253*) and ECC configuration.

The NAND controller supports the following features:

- Programmable page size of 256, 512, 1024, 2048, 4096 or 8192 bytes
- Programmable memory timings
- Programmable Error Correction Capability using BCH code or Hamming code
- Command sequencer: a hardware accelerator for read/write accesses within a page or within a block of configurable size.
  This feature is useful when ECC computation is performed.
- Programmable address offset for ECC bytes programming using the command sequencer
- Interrupt generation on ECC error detection and page or sector transfer complete
- Support of multiple dice per package. Only one die can be selected at a time.

Table 253. Programmable NAND flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory wait	Minimum duration (in fmc_ker_ck clock cycles) of the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256

Table 253. Programmable NAND flash access parameters (continued)

Parameter	Function	Access mode	Unit	Min.	Max.
Memory hold	Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory databus high-Z	Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started	Write	FMC clock cycle (fmc_ker_ck)	1	256

32.11.1 External memory interface signals

The following tables list the typical signals that are used to interface NAND flash memory.

Note: The "N" prefix identifies the signals which are active low.

8-bit NAND flash memory

Table 254. 8-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCEx	O	Chip Select (x=1, 2, 3, 4)
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
RNB	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 255. 16-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCEx	O	Chip Select (x=1, 2, 3, 4)
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)

Table 255. 16-bit NAND flash memory (continued)

FMC signal name	I/O	Function
NWE	O	Write enable
RNB	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

32.11.2 NAND flash supported memories and transactions

Table 256 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 256. Supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
	Asynchronous	R	64	8	Y	Split into 8 FMC accesses
	Asynchronous	W	64	8	Y	Split into 8 FMC accesses
NAND 16-bit	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

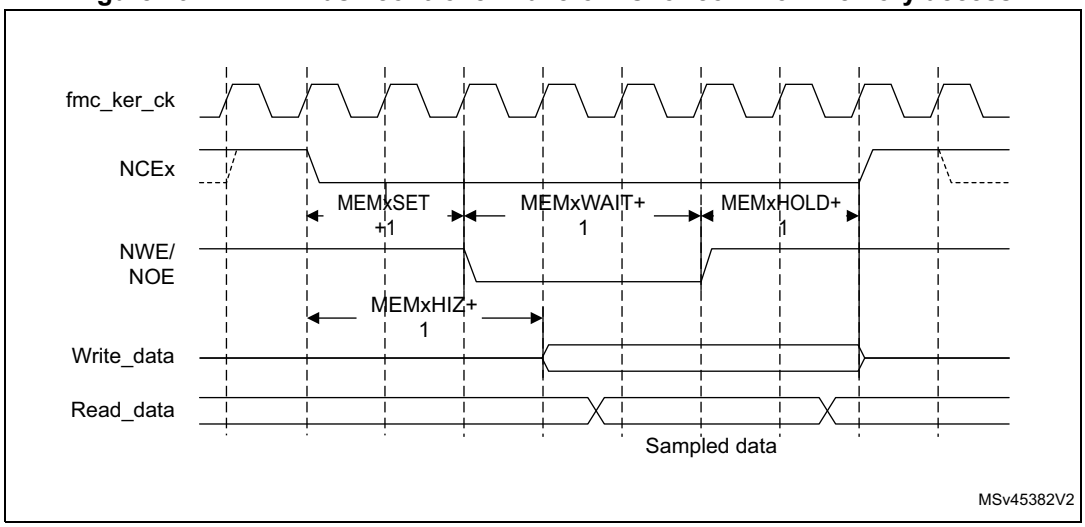
32.11.3 Timing diagrams for NAND flash memory

The NAND flash memory region is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_HECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

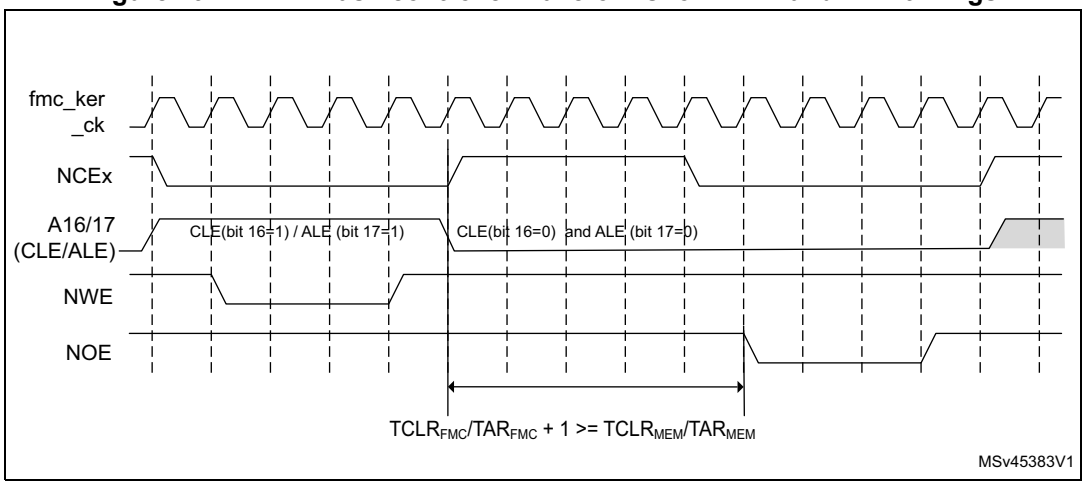
Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the different phases of any NAND flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. *Figure 191* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 191. NAND flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. If PWAITEN = 1, the programmed values of (MEMxSET + MEMxHOLD) must be > 1.

Figure 192. NAND flash controller waveforms for TCLR and TAR timings



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.

32.11.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the NAND memory region by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 32.11.5: NAND flash prewait function*) registers according to the characteristics of the NAND flash memory (PWID bits to configure NAND flash data bus width, PWAITEN set or cleared as needed, see *Section 32.7.2: NAND flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with a data byte value equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The NAND flash CLE input is active during the write strobe (low pulse on NWE), so that the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address for a read operation by writing a five-byte address to the common memory or the attribute space followed by the second read command. The NAND flash ALE input is active during the write strobe (low pulse on NWE), so that the written bytes are interpreted as the start address for read operations. Using the attribute memory space allows to use a different FMC timing configuration, which can be used to implement the prewait function required for some NAND flash memories (see details in *Section 32.11.5: NAND flash prewait function*).
4. The controller waits till the NAND flash memory is ready (RNB signal high), before starting a new access to the same memory region or to another memory region.
5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.
6. No CPU command or address write operation is required to read the next NAND flash page. This can be done in three different ways:
   - by simply performing the operation described in step 5,
   - a new random address can be accessed by restarting the operation at step 3,
   - a new command can be sent to the NAND flash device by restarting at step 2.

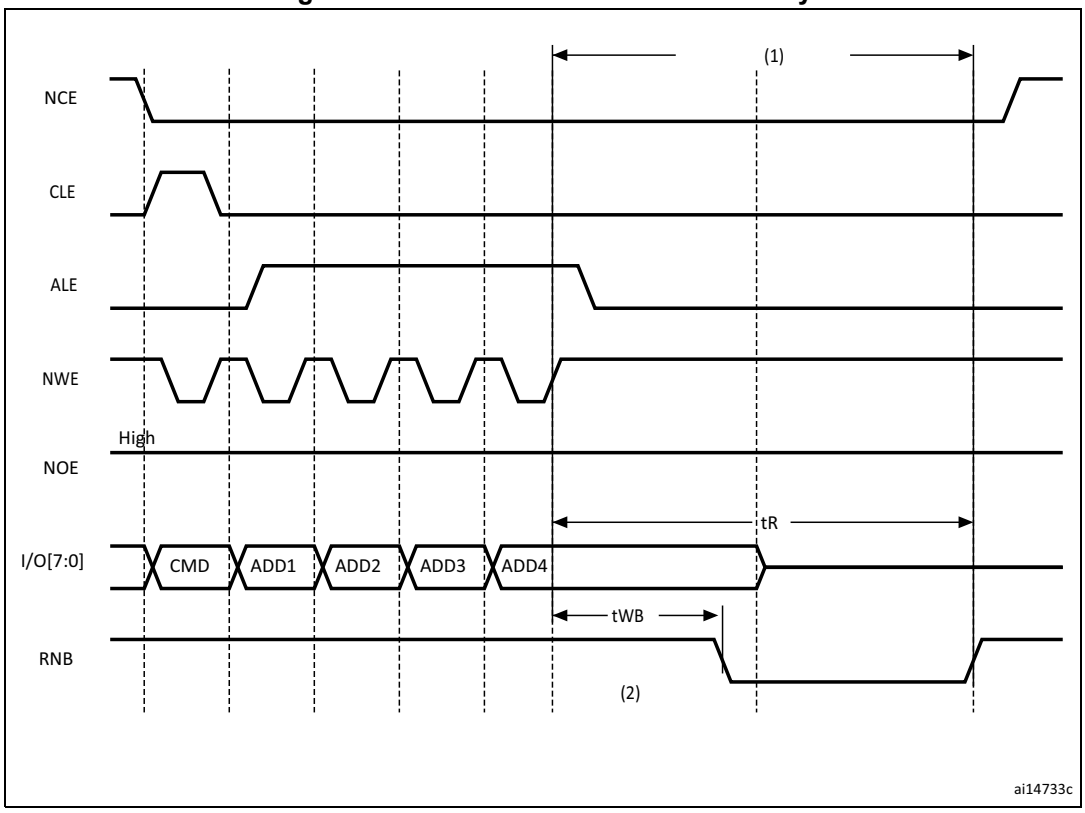
Note: Non-'CE don't care' NAND flash memories are not supported.

ONFI mode 4 and 5 EDO data output cycle timings are not supported.

32.11.5 NAND flash prewait function

The NAND controller must wait for RNB to be asserted low before sending its next command (see *Figure 193*). This timing constraint is set according to t_{WB} using MEMHOLD[7:0] bitfield in FMC_PMEM or ATTHOLD[7:0] in FMC_PATT.

Figure 193. Access to NAND flash memory



1. NCE can be deasserted for NAND flash memories since CE is don't care except for very old NAND flash devices that are not considered in this example.
2. RNB sampling can be delayed after t_{WB} by using ATTHOLD[7:0] bit of FMC_PATT. As an example, if fmc_ker_clk = 14 ns and ATTHOLD[7:0] = 7, the delay is 8*14 = 112 ns. The usual t_{WB} maximum value is 100 ns and the programmed delay must be higher than t_{WB} + 2*fmc_ker_clk.

Any CPU read or write access to the NAND flash memory has a delay of (MEMHOLD[7:0] + 1) * fmc_ker_ck cycles after the NWE signal is asserted.

The ATTHOLD[7:0] bitfield in FMC_PATT register must be determined to satisfy t_{WB} timing constraints and selectively apply larger timing constraints to the last command or address byte in CPU mode. When setting ATTHOLD[7:0], the 2 * fmc_ker_ck delay required for the sampling of RNB by NAND sequencer must also taken into account.

32.11.6 NAND ECC controller

NAND flash memories include a spare memory area (or OOB) at the end of each page. This area is typically used to store some metadata (such as Bad Block Marker information to invalidate the whole page) or for error management functions.

The FMC supports on-the-fly error correction code (ECC) computation during data read/program from/to NAND flash memory. This mechanism reduces the host CPU workload when processing the ECC by software.

The user application can select one out of two algorithms:

- Hamming code algorithm for 1-bit error code correction and 2-bit error detection per 256, 512, 1024, 2048, 4096 or 8192 bytes.
- BCH (Bose, Chaudhuri and Hocquenghem) code for 4-bit error code correction and 8-bit error detection or 8-bit error code correction and 16-bit error detection per 512 bytes.

The principle of both ECC controllers is similar: the ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory region is active. The ECC controllers operate as follows:

- When accessing a NAND flash memory region, the data present on the D[7:0] or D[15:0] bus are latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU or DMA, the ECC result registers are updated.

Hamming code

The Hamming code implemented can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4 096 or 8192 bytes read or written from/to the NAND flash memory. It consists in calculating the row and column parity. This algorithm is supported by 8-bit and 16-bit NAND flash memories.

Page write and read sequences with Hamming code are described below.

To perform a page program with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC controller computes the ECC value.
3. Wait until the ECC code is ready (NWRF is set in the FMC_SR register).
4. Read the ECC value available in the FMC_HECCR register and store it in NAND flash spare area.
5. Clear the ECCEN bit and launch the page programming.

To perform a page read with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register before reading the NAND page. While the NAND page is read, the ECC block computes the ECC value.
2. Wait until the ECC code is ready (NWRF set in the FMC_SR register).
3. Read the ECC value available in the spare area as well also the ECC value, processed by Hamming controller, available in FMC_HECCR register.
4. If the two ECC values are identical, no correction is required. Otherwise an ECC error occurred, the software correction routine returns information on whether the error can be corrected or not, and corrects it when possible.

Depending on the spare area size, the page can be split into several sectors of identical size. This allows increasing the error correction capability for each page since 1 error can be corrected and 2 errors can be detected for each sector.

Example

For a NAND flash memory with 2 Kbytes + 64 bytes pages, each page can be split into 8 sectors of 256 bytes each. 3 bytes (22 bits) of ECC are required for each sector, which makes 24 bytes (8 * 3 bytes) to be stored in the spare area for the whole page. For the whole 2 Kbyte page, up to 8 errors can be corrected, respectively up to 16 errors detected, provided there is less than 1 error, respectively 2 errors, per sector.

BCH (Bose, Chaudhuri and Hocquenghem) code

To increase the error correction capability, the FMC embeds a BCH (Bose, Chaudhuri and Hocquenghem) encoder and decoder.

It supports 4-bit error code correction and 8-bit error detection or 8-bit error code correction, and 16-bit error detection per sector. BCH encoder/decoder module handles sectors of fixed size, equal to 512 bytes. Each page must consequently be split into 512-byte sectors. The number of parity bytes generated for each sector is given in *Table 257*. For 16-bit NAND flash memories, a padding is added to align parity bytes to an even number (8 bytes and 14 bytes for 4-bit and 8-bit BCH code, respectively).

Table 257. Number of ECC parity bytes per sector

Function	BCH code	
Error correction	4-bit	8-bit
Number of parity bytes	7 bytes	13 bytes

The BCH module detects errors and returns the bit position for each sector. To correct the errors, the software needs to flip the error bits at the provided bit position stored in the FMC_BCHPBRx register.

The BCH ECC controller supports 8-bit and 16-bit NAND flash memories.

During a page program operation, the BCH encoder generates the ECC bytes (7 or 13 bytes) for each sector of 512 bytes that must be written in the NAND flash spare area with a programmable offset. It must take into account some metadata at the beginning of the spare area. This operation must be executed for each sector. The BCH encoder does not process the spare area: metadata must have their own error protection mechanism.

The total number of ECC bytes (N) is given by the below equation:

$$N = \text{Number of ECC bytes /sector} \times \text{Number of sector/page}$$

Example

For an 8-bit NAND flash with 4096-byte pages and an error correction capability of 4-bit per 512 bytes, the number of sector per page is 8 (4096 / 512). The total number of ECC bytes is consequently:

$$N = 7 \times 8 = 56 \text{ bytes}$$

The user application has to make sure that the spare area is large enough to store the metadata and parity bytes.

Page program and read sequences with BCH code are described below.

To perform a page program:

1. Enable write access by setting the WEN bit in the FMC_PCR register.
2. Enable the ECC by setting the ECCEN bit in FMC_PCR register.
3. Write 512 data bytes to the sector in the NAND flash memory page. While the sector is written, the BCH encoder computes the ECC value.
4. Wait until the BCH encoder parity bits are available by monitoring EPBRF bit of FMC_BCHISR register.
5. Read the ECC value available in the FMC_BCHPBRx registers and store it in the NAND flash spare area.
6. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
7. Execute again step 2 to 6 for all sectors to be written in NAND flash memory.
8. Launch the page programming.

To perform a page read:

1. Enable read access by resetting the WEN bit in the FMC_PCR register.
2. Enable the ECC by stetting the ECCEN bit in FMC_PCR register.
3. Read the NAND page sector (512 bytes).
4. Read the corresponding ECC bytes (7 or 13 bytes) from the NAND flash spare area.
5. During the read of data and parity bits, the syndrome is calculated. Then the error location search is launched.
6. Wait until the decoding error is ready (DERF bit set in the FMC_BCHISR register), then read the decoding results (errors detected if any, correctable or not, number of errors, bit position) in FMC_BCHDSR0 and FMC_BCHDSRx registers.
7. Thanks to the decoding results, the software can correct the errors in the sector (when possible).
8. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
9. Execute again step 2 to 7 for all relevant sectors of the page.

32.11.7 FMC command sequencer

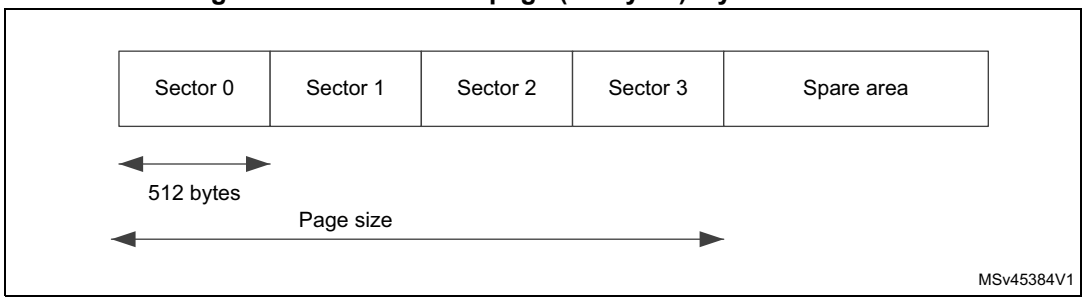
NAND flash page programming and reading can be performed by the CPU as described in the above sections.
Data sector programming/reading as well as ECC byte programming/reading can also be managed automatically by the FMC command sequencer for command and address transfers, and by DMA for data transfers.

The purpose of the command sequencer is to ease the programming/reading of NAND flash pages with ECC computation as well as free the CPU of sequencing tasks. The command sequencer enables the software to work on full NAND flash pages instead of sectors, which is more efficient and essential to achieve high throughput. The sequencer handles all the sectors in the page. It does not support sub-page read or program operations.

One DMA channel is required for write operations and two DMA channels for read operations.

A specific NAND flash page layout must be respected when using the BCH algorithm. See *Figure 194* for an example of 2Kbyte page BCH layout.

Figure 194. NAND flash page (2 Kbytes) layout with BCH



Page program sequence

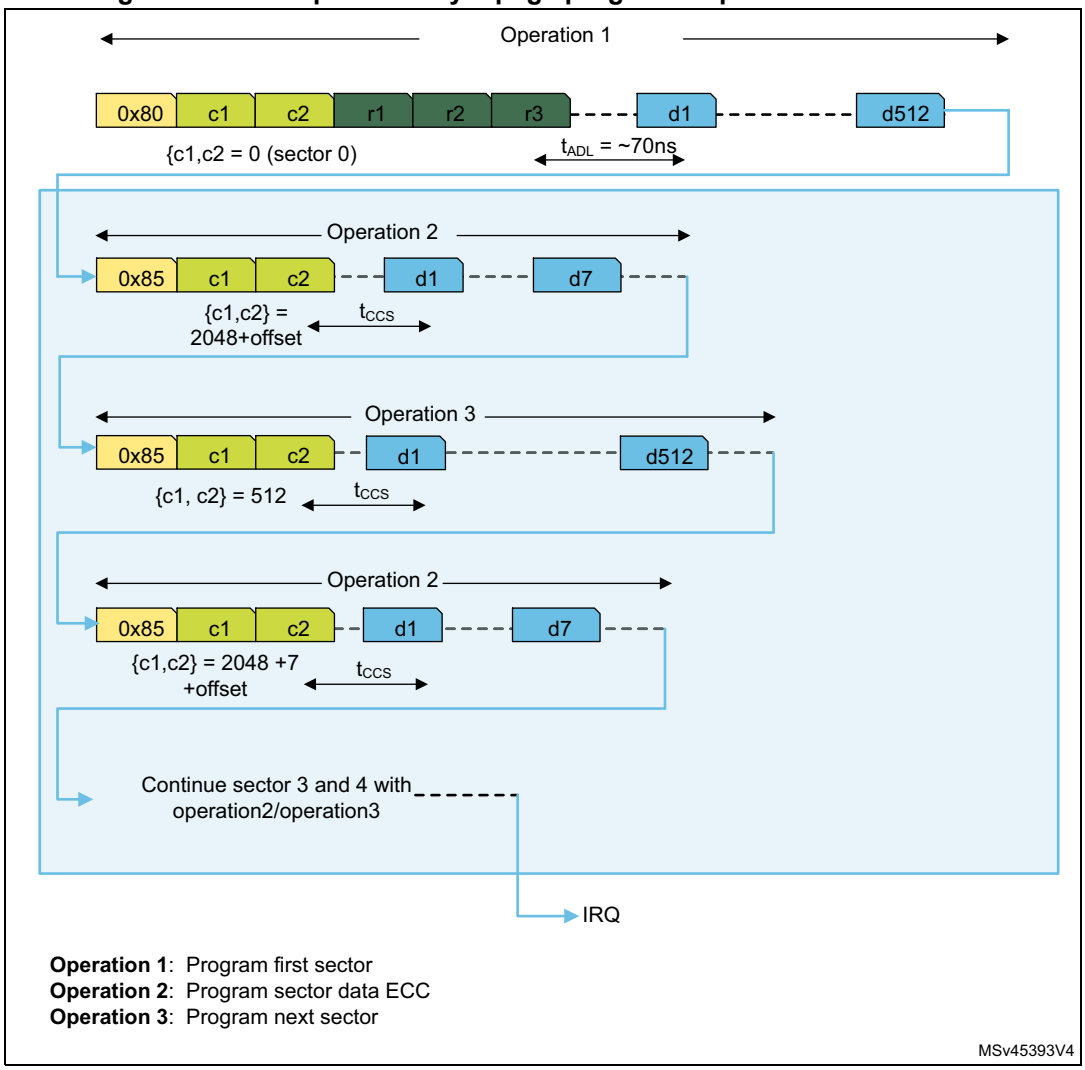
1. A data buffer located in SRAM has to be initialized with the data to be written to the NAND flash page.
2. The CPU configures the FMC command sequencer registers: page program command (0x80), address bytes, change write column command (0x85), ECC address offset, page size, sector size and sector number. It also selects the chip enable.

   It also selects the ECC algorithm (Hamming or BCH) and configures one DMA channel to transfer data from the buffer to the NAND flash page.
3. The CPU then launches the sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The command sequencer automatically generates the control and address cycles to the NAND flash memory, following the programmed timings configured either in FMC_PMEM or FMC_PATT register.
5. A DMA request is issued to start data transfer from the data buffer to NAND flash memory.
6. When the full sector is programmed, go to step 7 if ECC is enabled. Otherwise, jump to step 9.
7. The FMC command sequencer automatically issues a change write column command (0x85) followed by two address cycles. The command to be issued and the number of address cycles are programmed in FMC_CSQCFGR2 register. The address is automatically generated from the programmed ECC address offset.
8. Once the parity bytes are available in ECC encoder (Hamming or BCH) module, they are automatically transferred to the NAND flash spare area. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
9. The sequencer then repeats step 4 (except that it performs only a change write column command instead of a page program command) to step 8 to program the following sectors and their corresponding ECC parity bytes. The ECC address offset is automatically incremented for each sector transfer. When ECC is disabled, the sequencer repeats step 5.
10. The FMC command sequencer can generate a transfer complete interrupt or a transfer complete error interrupt by sector or when all sectors are written.
11. The CPU can program any further metadata in the NAND flash area, taking care not to overwrite parity bytes.
12. The program command (0x10) can then be written to the command section to program the NAND flash memory.

13. An interrupt, if enabled, is generated on the rising edge of the Ready/Busy (RNB) signal to indicate that the page program operation is complete.

Refer to *Figure 195* for an overview of the page program sequence with BCH.

Figure 195. Example of 2 Kbyte page program sequence with 4-bit BCH



Page read sequence

1. A data buffer is allocated in SRAM to store the NAND flash page data. A decoding status buffer is also allocated to store result of ECC decoding.
2. The CPU programs the command sequencer registers: the page read first cycle command (0x00), address bytes, the page read second cycle command (0x30), change read column command (0x05), spare area offset, page size, sector size and sector number. It also selects the chip enable.

   It configures the ECC algorithm if required (Hamming or BCH), one DMA channel to handle data transfers from the NAND flash page to the data buffer, and a second DMA

channel to transfer ECC decoding results from ECC status registers to the RAM decoding status buffer.

3. The CPU then launches the FMC sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The FMC command sequencer issues the command and address cycles to the NAND flash memory.
5. The command sequencer waits for a Ready/Busy (RNB) rising edge to start data transfer.
6. It sends a DMA request to launch the data transfer from the NAND flash page to the data buffer.
7. When all sector data are read, go to step 8 if ECC is enabled, otherwise go to step 9. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
8. When ECC status registers are empty (because the Hamming error code or the BCH error position bits of previous sector have already been read), the ECC command sequencer automatically issues a change read column command (0xE0) followed by two address cycles and by the 0xE0 command, to read ECC bytes from NAND flash memory and store them in ECC registers. These commands as well as the number of address cycles are programmed in the FMC_CSQCFGR2 register (see *Figure 196*).

   The address in spare area is extracted from the programmed ECC address offset.

   When the Hamming algorithm is used, a DMA request to the second DMA channel is send to transfer ECC codes (FMC_HPR and FMC_HECCR registers) to the decoding status buffer.

   When BCH algorithm is used, once all the parity bytes have been read, the BCH decoder first processes the syndrome to detect errors and then searches for error positions if any.

   As the BCH decoder is pipelined, it is possible to start reading a sector even if the decoding of the previous sector is not complete.

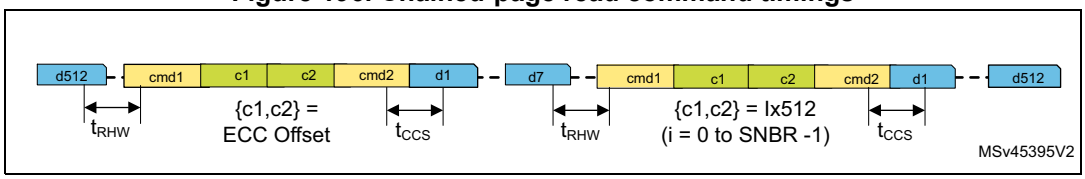
   In parallel of the following steps, when BCH sector decoding is complete, the new error status is stored in the ECC status registers (FMC_BCHDSR0) and a DMA request to the second DMA channel is sent to transfer data from the status registers to the decoding status buffer located in RAM.
9. The FMC command sequencer then repeats step 4 (except that a change read column command is issued instead of the page read first cycle command) to step 8 to read the other sectors and their corresponding ECC bytes. The addresses are issued and incremented following the page size, number of sectors, spare area address offset address and BCH code selection. When the read sequence is complete, the command sequencer waits for t_{RHWL} before issuing a new command sequence (refer to *Figure 195*). When ECC is disabled, the sequencer repeats step 6.
10. Once all sectors within a page and the status registers (FMC_BCHDSRx) are transferred by the command sequencer, the ECC controller can generate the transfer complete interrupt or transfer complete error (when enabled).
11. By reading the decoding status buffer, the CPU can start processing the errors corresponding to each sector. If no error has been detected, the page stored in RAM is validated. Otherwise the CPU corrects the error(s) before validating the page buffer in RAM. If there are too many errors, the CPU can detect the page cannot be corrected and invalidate the data buffer.

Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and the DUE bit is set. If the number of errors the detected by the BCH decoder exceeds the maximum number of detectable errors, DEN and DUE bits are not significant.

The DMA channels can also send interrupts each time a data transfer is complete. This allows the CPU to process error(s) in a sector as soon as it is available.

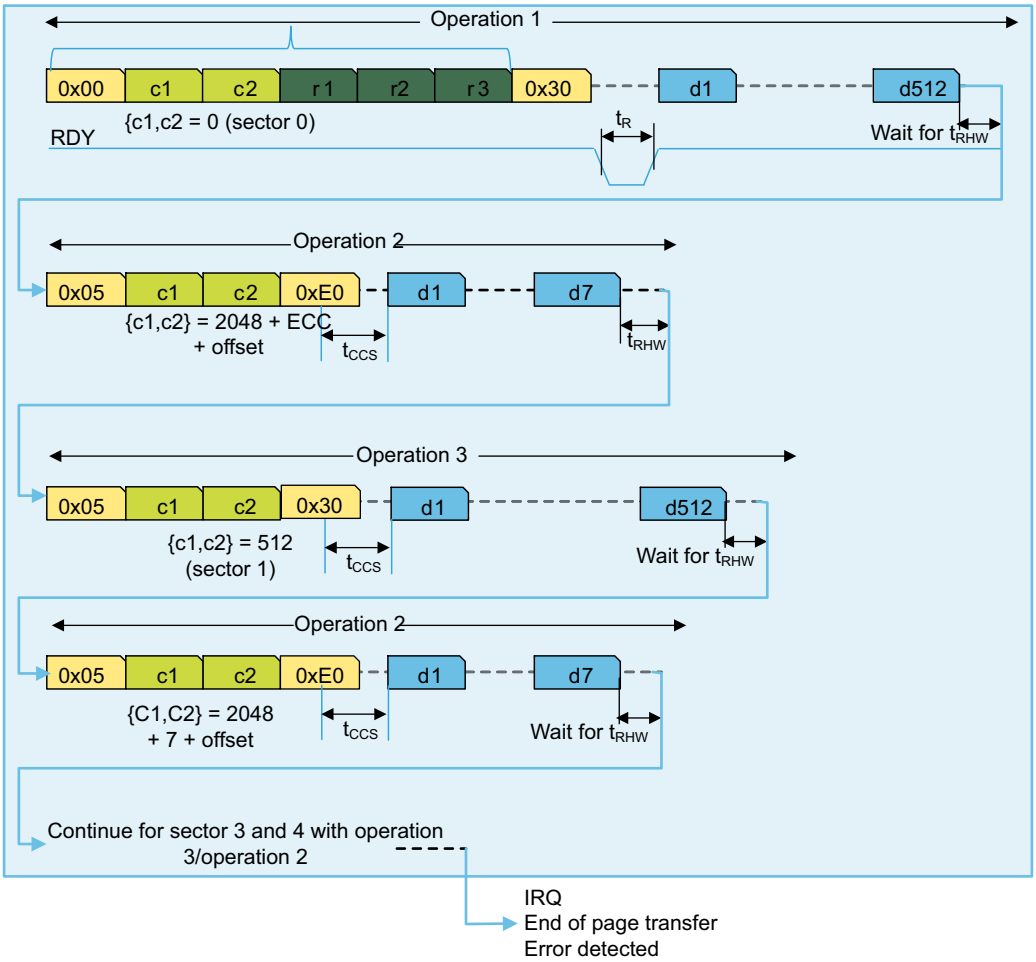
Note: When using the sequencer and DMA channels, the CPU must not read the FMC_BCHPBRx and FMC_BCHDSRx registers before data of all sectors are transferred, that is when TCF flag is set in the FMC_CSQISR register.

Figure 196. Chained-page read command timings



An example of 2 Kbyte page read sequence with BCH is described in *Figure 197*. This example corresponds to four 512-byte sectors and 7 ECC bytes per data 512 bytes.

Figure 197. Example of 2-Kbyte page read sequence with 4-bit BCH



The BCH controller resets the ECC computation whenever a page read (0x00), a page program (0x80), a random read (0x05) or a random program command ((0x85) is issued.

When only part of a page is written/read using the BCH, the command sequencer issues a random read/write command sequence only for the number of sectors to be read/written. This is done by programming the number of sector to be processed.

FMC command sequencer configuration

1. Configure the FMC_CSQCFGR1 register: program the command1 and the command 2 (in case of read access), the number of address cycles to be issued, configure the read or write sequence, and select the timings to be used.
2. Configure the address bytes in FMC_CSQAR1 and FMC_CSQAR2 register.
3. In the FMC_CSQCFGR2 register, configure the change write or change read column commands and the timings to be used for ECC. Program the address offset for the random columns in the FMC_CSQAR2 register. Enable the DMA if it is needed for ECC status register transfer.
4. Configure the number of sectors and the address cycle timings in the FMC_CSQCFGR3 register.
5. Configure the ECC sector size, and select read or write access in the FMC_PCR register.
6. Enable the SQSDTEN bit in the FMC_CSQFCGR2 register. The ECCEN bit in the FMC_PCR register kept at 0.
7. Transfer complete, and transfer error complete or sector transfer complete can be enabled through the FMC_CSQIER register.

32.11.8 NAND flash controller interrupt

An interrupt can be generated on the following events:

- RNB rising edge: an interrupt is generated when a rising edge is detected on RNB NAND flash signal
- RNB falling edge: an interrupt is generated when a falling edge is detected on RNB NAND flash signal
- RNB high-level: an interrupt is generated when a high-level is detected on RNB NAND flash signal.
- Transfer complete: an interrupt is generated when the FMC command sequencer has completed all data transfers for all sectors or when the command sequencer was aborted.
- Sector complete interrupt: an interrupt is generated when the FMC command sequencer has completed all data transfers for one sector.
- Sector error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer with at least one error detected.
- Sector uncorrectable error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer and detected an uncorrectable error.
- Command transfer complete: the command sequencer has completed the transfer of programmed commands and addresses.
- BCH encoder parity bits ready: an interrupt is generated when the BCH encoder has finished processing parity bits.
- BCH decoder syndrome ready: an interrupt is generated when the BCH decoder has finished processing the syndrome and knows if errors have been detected or not.
- BCH decoder error found: an interrupt is generated when the BCH decoder has detected an error.
- BCH decoder error ready: an interrupt is generated when the BCH decoder has finished processing errors, has identified error locations and can then correct them.
- BCH decoder uncorrectable error: an interrupt is generated when the BCH decoder has detected that the errors cannot be corrected (more than four or eight errors according to BCH error correction capability configuration).

The interrupt source can be enabled or disabled separately through the FMC_ IER, FMC_CSQIER and FMC_BCHIER registers.

Table 258. FMC NAND controller Interrupt request

Interrupt event	Event Flag	Enable control bit
Transfer complete	TCF in FMC_CSQISR	TCIE in FMC_CSQIER
Sector complete	SCF in FMC_CSQISR	SCIE in FMC_CSQIER
Sector error	SEF in FMC_CSQISR	SEIE in FMC_CSQIER
Sector uncorrectable error	SUEF in FMC_CSQISR	SUEIE in FMC_CSQIER
Command transfer complete	CMDTCF in FMC_CSQISR	CMDTCIE in FMC_CSQIER
BCH encoder parity bits ready	EPBRF in FMC_BCHISR	EPBRIE in FMC_BCHIER
BCH decoder syndrome ready	DSRF in FMC_BCHISR	DSRIE in FMC_BCHIER
BCH decoder error found	DEFF in FMC_BCHISR	DEFIE in FMC_BCHIER

Table 258. FMC NAND controller Interrupt request (continued)

Interrupt event	Event Flag	Enable control bit
BCH decoder error ready	DERF in FMC_BCHISR	DERIE in FMC_BCHIER
BCH decoder uncorrectable error	DUEF in FMC_BCHISR	DUEIE in FMC_BCHIER

32.11.9 NAND flash controller registers

NAND flash programmable control register (FMC_PCR)

Address offset: 0x80

Reset value: 0x0007 FE08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR[3]
						rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				ECCALG	Res.	ECCEN	PWID[1:0]		Res.	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **WEN:** Write enable

This bit enables read or write access. It must be configured when using the FMC sequencer or when the ECC with BCH code is enabled. It can only be modified when ECCEN bit is reset.

0: Read access enabled

1: Write access enabled

Bit 24 **BCHECC:** BCH error correction capability

0: 4-bit BCH (4-bit error correction and 8-bit error detection per 512 bytes)

1: 8-bit BCH (8-bit error correction and 16-bit error detection per 512 bytes)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCSS[2:0]:** ECC sector size (used to access spare area)

These bits define the sector size used for the ECC:

000: 256 bytes

001: 512 bytes

010: 1024 bytes

011: 2048 bytes (default)

100: 4096 bytes

101: Reserved.

11X: Reserved.

When the ECC is enabled with BCH code, the sector size must be programmed to 512 bytes.

Bits 16:13 **TAR[3:0]:** ALE to RE delay.

These bits define the time from ALE low to RE low in number of fmc_ker_ck clock cycles:
t_ar ≤ (TAR +1) × $t_{fmc_ker_ck}$
where $t_{fmc_ker_ck}$ is the FMC clock period and *t*_ar is the ALE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bits 12:9 **TCLR[3:0]:** CLE to RE delay.

These bits define the time from CLE low to RE low in number of fmc_ker_ck clock cycles:
t_clr ≤ (TCLR +1) × $t_{fmc_ker_ck}$
where $t_{fmc_ker_ck}$ is the FMC clock period and *t*_clr is the CLE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bit 8 **ECCALG:** ECC algorithm

0: Hamming code is selected (default).
1: BCH code is selected.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit

0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.
This bit must be kept reset when using the FMC sequencer

Bits 5:4 **PWID[1:0]:** Data bus width

These bits define the external memory device width.
00: 8 bits (default after reset).
01: 16 bits
1X: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PBKEN:** NAND flash memory region enable bit

This bit enables the memory region. Accessing a disabled memory region causes an AXI bus error.
0: Corresponding memory region is disabled (default after reset)
1: Corresponding memory region is enabled.

Note: The NAND flash memory region must be disabled only when all NAND flash controller commands are complete.

Bit 1 **PWAITEN:** Wait feature enable bit

This bit enables the Wait feature for the NAND flash memory region:
0: disabled (default)
1: enabled

Bit 0 Reserved, must be kept at reset value.

FMC status register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0053

This register contains information about the AXI interface isolation status and the NAND write requests status. The FMC has to be disabled before modifying some registers. As requests might be pending, it is necessary to wait till the AXI interface is stable and the core of the block is totally isolated from its AXI interface before reconfiguring the registers.

The PEF and NWEF bits indicate the status of the pipe. If Hamming algorithm is used, the ECC is calculated while data are written to the memory. To read the correct ECC, the software must consequently wait until no write request to the NAND controller are pending, by polling PEF and NWRF bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWRF	Res.	PEF	Res.	Res.	ISOST[1:0]	
									r		r			r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **NWRF:** NAND write request flag

This bit provides the status of the write request issued to the NAND flash memory. When this bit is set, all write requests to the NAND controller are served.

0: NAND flash write requests are pending

1: No NAND flash write requests pending

Bit 5 Reserved, must be kept at reset value.

Bit 4 **PEF:** Pipe Empty Flag

This bit indicates if the pipe contains requests.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ISOST[1:0]:** FMC isolation state with respect to the AXI interface

00: FMC is not isolated; AXI transactions are treated by the FMC and its controllers.

01: FMC has been enabled (FMCEN = 1) and waits for the end of pending AXI transactions (which leads to error responses) before going to “non-isolated” state.

10: FMC has been disabled (FMCEN = 0) and waits for the end of pending AXI transactions before going to isolation state.

11: FMC is isolated from its AXI interface. All AXI requests lead to error responses.

FMC common memory space timing register (FMC_PMEM)

Address offset: 0x88

Reset value: 0x0A0A 0A0A

The FMC_PMEM read/write register contains NAND flash memory region timing information. This information is used to access the NAND flash common memory space for command, address write accesses or data read/write accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEMHIZ[7:0]								MEMHOLD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMWAIT[7:0]								MEMSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MEMHIZ[7:0]:** Common memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept Hi-Z after starting a NAND flash write access to the common memory space. This is only valid for write transactions:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycles
.....
1111 1111: 256 * fmc_ker_ck cycles

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), during NAND flash read or write accesses to the common memory space:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycles
.....
1111 1111: 256 * fmc_ker_ck cycles

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

These bits define the minimum number of fmc_ker_ck clock cycles (+1) to assert the command (NWE, NOE), during NAND flash read or write accesses to the common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: 1 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
.....
1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **MEMSET[7:0]:** Common memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) to set up the address before the command assertion (NWE, NOE), during NAND flash read or write accesses to the common memory space:
0000 0000: 1 * fmc_ker_ck cycles
0000 0001: 2 * fmc_ker_ck cycles
.....
1111 1111: 256 * fmc_ker_ck cycles

FMC attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0x0A0A 0A0A

The FMC_PATT read/write register contains NAND flash memory region timing information. It is used for 8-bit accesses to the NAND flash attribute memory space during the last address write access when the timing differs from previous accesses (for Ready/Busy management, refer to *Section 32.11.5: NAND flash prewait function*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATTHIZ[7:0]								ATTHOLD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTWAIT[7:0]								ATTSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept in Hi-Z after starting a NAND flash write access to the attribute memory space. These bits are only valid for write transactions:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write access) after the command deassertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), during NAND flash read or write accesses to the attribute memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: 1 * fmc_ker_ck cycle (+ wait cycle introduced by deassertion of NWAIT)
0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deassertion of NWAIT)
.....
1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) for address set up before the command assertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles.

FMC Hamming parity result registers (FMC_HPR)

Address offset: 0x90

Reset value: 0x0000 0000

This register is used during read accesses in conjunction with the FMC sequencer. It contains the current error correction code value computed by the FMC NAND controller Hamming module. When the FMC sequencer reads data from a NAND flash memory page at the correct address, the data read are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area and stored in the and the FMC_HPR, to determine whether a page is valid, and to correct it otherwise. The FMC_HPR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HPR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HPR[31:0]:** Hamming parity result

This field contains the parity value computed by the Hamming ECC computation module. *Table 259* describes the contents of these bitfields.

Table 259. HPR relevant bits

ECCSS[2:0]	Sector size in bytes	HPR bits
000	256	HPR[21:0]
001	512	HPR[23:0]
010	1024	HPR[25:0]
011	2048	HPR[27:0]
100	4096	HPR[29:0]
101	8192	HPR[31:0]

FMC Hamming code ECC result register (FMC_HECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the FMC NAND controller Hamming module.When the CPU reads/writes data from/to a NAND flash memory page at the correct address (refer to *Section 32.11.6: NAND ECC controller*), the data read/written from/to the NAND flash memory are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS

field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and to correct it otherwise. The FMC_HECCR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HECC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HECC[31:0]:** ECC result

This field contains the value computed by the Hamming ECC computation logic. *Table 260* describes the contents of these bitfields.

Table 260. ECC result relevant bits(1)

ECCSS[2:0]	Sector size in bytes	HECC bits
000	256	HECC[21:0]
001	512	HECC[23:0]
010	1024	HECC[25:0]
011	2048	HECC[27:0]
100	4096	HECC[29:0]
101	8192	HECC[31:0]

1. Only the relevant bits according to the sector size are valid.

FMC NAND interrupt enable register (FMC_IER)

Address offset: 0x180

Reset value: 0x0000 0000

This register configures the interrupt capability generation when the RNB signal is connected on FMC interrupt pin. An interrupt can be generated on the rising edge, falling edge or high level detected on the RNB signal.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEE	IHLE	IREE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IFEE:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 1 **IHLE:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 0 **IREE:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

FMC interrupt status register (FMC_ISR)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEF	IHLF	IREF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IFEF:** Interrupt falling edge flag
The flag is set by hardware and cleared by setting the CFEIF bit.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

Bit 1 **IHLF:** Interrupt high-level flag
The flag is set by hardware and cleared by setting the CHLIF bit.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IREF:** Interrupt rising edge flag
The flag is set by hardware and cleared by setting the CREIF bit.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

FMC NAND controller interrupt clear register (FMC_ICR)

Address offset: 0x188

Reset value: 0x0000 0000

Writing 1 in one of these bits clears the corresponding flag in the interrupt status register (FMC_ISR).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIFEF	CIHLF	CIREF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CIFEF:** Clear Interrupt falling edge flag
0: No effect
1: Clears the IFEF flag in the FMC_ISR register

Bit 1 **CIHLF:** Clear Interrupt high-level flag
0: No effect
1: Clears the IHLF flag in the FMC_ISR register

Bit 0 **CIREF:** Clear Interrupt rising edge flag
0: No effect
1:Clear the IREF flag in the FMC_ISR register

FMC NAND command sequencer control register (FMC_CSQCR)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSQSTART
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CSQSTART**: Command Sequencer Enable
Writing 1 starts the Command sequencer.
Writing 0 has no effect.

Note: Write this bit to 1 to start the sequencer only when the FMC is enabled by setting FMCEN bit of FMC_CFGR to 1 and the NAND flash controller is activated by setting PBKEN bit of FMC_PCR to 1.

FMC NAND command sequencer configuration register 1 (FMC_CSQCFGR1)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD1[7:0]								Res.	ACYNBR[2:0]			Res.	DMADEN	CMD2EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CMD2T:** Command 2 Sequencer timings

This bit specifies if the Command 2 (CMD2) is issued to the NAND flash memory with the timings defined in FMC_PMEM or FMC_PATT register.

0: CMD2 issued with the timings programmed in FMC_PMEM

1: CMD2 issued with the timings programmed in FMC_PATT

Bit 24 **CMD1T:** Command 1 Sequencer timings

This bit specifies if the Command 1 (CMD1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **CMD2[7:0]:** Command 2 sequencer

These bits specify if the Command 2 is issued by the command sequencer for the command second cycle during read operations.This command is only issued when the CMD2EN bit is set.

Example: CMD2[7:0] bits must be programmed to 0x30 to issue the read column command.

Bits 15:8 **CMD1[7:0]:** Command 1 sequencer

These bits specify if the Command 1 is issued by the command sequencer for the first cycle during read or write operations.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACYNBR[2:0]:** Address Cycle number

These bits define the number of address cycles to be generated by the command sequencer when issuing the command programmed in the registers.

000: No address cycle.

001: 1 address cycle

010: 2 address cycles

011: 3 address cycles

100: 4 address cycles

101: 5 address cycles

Others: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DMADEN:** Command sequencer DMA request data enable

When this bit is set, the command sequencer generates a DMA request to the first DMA channel, to transfer data from/to NAND flash memory.

0: No DMA request transfer

1: A DMA request transfer

Bit 1 **CMD2EN:** Command cycle 2 Enable

This bit specifies that the command cycle 2 has to be generated by the command sequencer.

0: Command cycle 2 not issued.

1: Command cycle 2 (programmed CMD2[7:0]) sent by the command sequencer to the NAND flash memory after the address cycles.

Bit 0 Reserved, must be kept at reset value.

FMC NAND command sequencer configuration register 2 (FMC_CSQCFGR2)

Address offset:0x208

Reset value: 0x0000 0000

This register is used to configure the command sequencer to issue random read/ write commands to read/ write data by sector and automatically read/write data from NAND flash memory at a programmable address offset. This is useful when performing a sector read/write operation followed by an ECC read/write operation in the NAND flash spare area.The command sequencer generates the random commands until all the sectors are read/written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCMD1[7:0]								Res.	Res.	Res.	Res.	Res.	DMASEN	RCMD2EN	SQSDTEN
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RCMD2T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bit 24 **RCMD1T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings flash memory in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **RCMD2[7:0]:** Random Command 2 sequencer

These bits specify that the command 2 is issued by the command sequencer for the second cycle during read operations.This command is only issued when the RCMD2EN bit is set.
Example: the RCMD2 (0xE0) is required to issue the random read column command.

Bits 15:8 **RCMD1[7:0]:** Random Command 1 sequencer

These bits specify that the command 1 value is issued by the command sequencer for the first cycle during read or write accesses.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **DMASEN:** Command sequencer DMA request decoding status enable

When this bit is set, the command sequencer generates a DMA request to the second DMA channel when valid bits are available in the ECC status register at the end of ECC processing (decoding).
0: No DMA request used for ECC status registers transfer
1: A DMA request used for ECC status registers transfer

Note: This bit must be set only during a page read sequence to offload the saving of the ECC results between sectors by using a second DMA channel.

Bit 1 **RCMD2EN:** Random Command 2 sequencer enable

This bit enables the command 2 to be issued by the command sequencer during read accesses. This command is issued only when the CMD2SQEN bit is set.
0: Command 2 not issued.
1: Command 2 (CMD2SQ[7:0]) issued by the command sequencer to NAND flash memory after the address cycle.

Bit 0 **SQSDTEN:** Sequencer spare data transfer enable

This bit enables the sequencer to access the spare data area after each sector transfer. It also enables the ECC when using the sequencer. ECCEN bit in FMC_PCR register must be reset.
0: ECC disabled and spare data area not accessed by the sequencer
1: ECC enabled and spare data area read or programmed by the sequencer after each sector transfer

FMC NAND sequencer configuration register 3 (FMC_CSQCFGR3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SNBR[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **RAC2T:** Random Address cycle 2 sequencer timings

This bit specifies if the random Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC2 issued with the timings programmed in FMC_PMEM

1: Random ADDC2 issued with the timings programmed in FMC_PATT

Bit 22 **RAC1T:** Random Address cycle 1 sequencer timings

This bit specifies if the random Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC1 issued with the timings programmed in FMC_PMEM

1: Random ADDC1 issued with the timings programmed in FMC_PATT

Bit 21 **SDT:** Spare data transfer sequencer timings

This bit specifies if the spare data transfer for read and write operations from/to the NAND flash memory are performed with the timings programmed in FMC_PMEM or FMC_PATT register.

0: Spare data transfer issued with the timings programmed in FMC_PMEM

1: Spare data transfer issued with the timings programmed in FMC_PATT

Bit 20 **AC5T:** Address cycle 5 sequencer timings

This bit defines if the Address cycle 5(ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC5 issued with the timings programmed in FMC_PMEM

1: ADDC5 issued with the timings programmed in FMC_PATT

Bit 19 **AC4T:** Address cycle 4sequencer timings

This bit defines if the Address cycle 4 (ADDC4) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC4 issued with the timings programmed in FMC_PMEM

1: ADDC4 issued with the timings programmed in FMC_PATT

Bit 18 **AC3T:** Address cycle 3 sequencer timings

This bit defines if the Address cycle 3 (ADDC3) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC3 issued with the timings programmed in FMC_PMEM

1: ADDC3 issued with the timings programmed in FMC_PATT

Bit 17 **AC2T:** Address cycle 2 sequencer timings

This bit defines if the Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC2 issued with the timings programmed in FMC_PMEM

1: ADDC2 issued with the timings programmed in FMC_PATT

Bit 16 **AC1T:** Address cycle 1 sequencer timings

This bit defines if the Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC1 issued with the timings programmed in FMC_PMEM

1: ADDC1 issued with the timings programmed in FMC_PATT

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **SNBR[5:0]:** Number of sectors to be read/written

These bits define the number of sectors that are processed by the sequencer.
They are used to read/write part of a page or to use the ECC (in particular with the BCH code which requires 512-byte sectors):
000000: 1 sector
000001: 2 sectors
...
111111: 16 sectors
Others: Reserved.

Bits 7:0 Reserved, must be kept at reset value.

FMC NAND command sequencer address register 1 (FMC_CSQAR1)

Address offset: 0x210

Reset value: 0x0000 0000

This register is used to define the value of address cycles 1 to 4 to be issued by the command sequencer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDC4[7:0]								ADDC3[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDC2[7:0]								ADDC1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ADDC4[7:0]:** Address Cycle 4

These bits define the value of address cycle 4 to be issued by the command sequencer during read or write accesses.

Bits 23:16 **ADDC3[7:0]:** Address Cycle 3

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 15:8 **ADDC2[7:0]:** Address Cycle 2

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 7:0 **ADDC1[7:0]:** Address Cycle 1

These bits define the value of address cycle 1 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer address register 2 (FMC_CSQAR2)

Address offset: 0x214

Reset value: 0x0000 0000

This register is used to program the fifth address cycle and the address offset in spare area. It also selects the chip enable.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NAND CEN1	NAND CEN0	Res.	Res.	ADDC5[7:0]							
				rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **SAO[15:0]:** Spare Area Address Offset

These bits define the offset of the first ECC byte in the spare area for the program sequence.

Note: The minimum SAQ value is 2 since the first two bytes of the spare area are reserved for Bad block marking and metadata.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **NANDCEN[1:0]:** NAND flash chip enable number

These bits define the chip enable to be selected by the NAND flash command sequencer.

00: NCE1 chip enable selected

01: NCE2 chip enable selected

10: NCE3 chip enable selected

11: NCE4 chip enable selected

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:0 **ADDC5[7:0]:** Address Cycle 5

These bits define the value of address cycle 5 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer interrupt enable register (FMC_CSQIER)

Address offset: 0x220

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCIE	SUEIE	SEIE	SCIE	TCIE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDTCIE:** Command Transfer Complete interrupt enable
0: Command Transfer Complete Interrupt disable
1: Command Transfer Complete Interrupt enable

Bit 3 **SUEIE:** Sector Uncorrectable Error interrupt enable
0: Command Transfer Complete Interrupt disable
1: Command Transfer Complete Interrupt enable

Bit 2 **SEIE:** Sector Error interrupt enable
0: Sector Error Interrupt disable
1: Sector Error Interrupt enable

Bit 1 **SCIE:** Sector Complete interrupt enable
0: Sector Complete Interrupt disable
1: Sector Complete Interrupt enable

Bit 0 **TCIE:** Transfer Complete Interrupt enable
0: Transfer Complete Interrupt disable
1: Transfer Complete Interrupt enable

FMC NAND command sequencer interrupt status register (FMC_CSQISR)

Address offset: 0x224

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCF	SUEF	SEF	SCF	TCF
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDTCF:** Command Transfer Complete flag
This bit is set when the command sequencer has completed the transfer of programmed commands and addresses.
This bit is cleared by setting CCMDTCF bit. An interrupt can be generated if the CMDTCIE bit is set in the FMC_CSQICR register.

Note: The Sector Transfer Complete interrupt is not needed in sequencer mode.

Bit 3 **SUEF:** Sector Uncorrectable Error flag
This bit is set by hardware and cleared by writing 1 to the CSUEF bit in FMC_CSQICR.
When this bit is set, it indicates that the command sequencer detected an uncorrectable error when decoding a sector. An interrupt can be generated if the SUEIE bit is set in the FMC_CSQICR register.

Note: The SUEF bit is relevant only for BCH code to indicate uncorrectable errors. It must be ignored for Hamming code.

Bit 2 **SEF:** Sector Error flag

This bit is set by hardware and cleared by writing 1 to the CSEF bit in FMC_CSQICR.

This bit is set when the command sequencer has completed the sector data transfer with at least one error detection. An interrupt can be generated if the SEIE bit is set in the FMC_CSQICR register.

Note: The SEF bit is relevant only for BCH code to indicate that errors are present. It must be ignored for Hamming code.

Bit 1 **SCF:** Sector Complete flag

This bit is set by hardware and cleared by writing 1 to the CSCF bit in FMC_CSQICR.

This bit is set when the command sequencer has completed the sector data transfer. An interrupt can be generated if the SCIE bit is set in the FMC_CSQICR register.

Bit 0 **TCF:** Transfer Complete flag

This bit is set by hardware and cleared by writing 1 to the CTCF bit in FMC_CSQICR.

This bit is set when the command sequencer has transferred all the data for all sectors or when the command sequencer was aborted. An interrupt can be generated if the TCIE bit is set in the FMC_CSQICR register.

FMC NAND command sequencer interrupt clear register (FMC_CSQICR)

Address offset: 0x228

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDTCF	CSUEF	CSEF	CSCF	CTCF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CCMDTCF:** Clear Command Transfer Complete flag

Writing one clears the CMTCF flag in the FMC_CSQISR register

Bit 3 **CSUEF:** Clear Sector uncorrectable Error flag

Writing one clears the SUEF flag in the FMC_CSQISR register

Bit 2 **CSEF:** Clear Sector Error flag

Writing one clears the SEF flag in the FMC_CSQISR register

Bit 1 **CSCF:** Clear Sector Complete flag

Writing one clears the SCF flag in the FMC_CSQISR register

Bit 0 **CTCF**: Clear Transfer Complete flag

Writing one clears the TCF flag in the FMC_CSQISR register

FMC command sequencer error mapping status register (FMC_CSQEMSR)

Address offset: 0x230

Reset value: 0x0000 0000

This register holds a sector error mapping status when the whole transfer is complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEM[15:0]:** Sector Error mapping
When SEM[i] bit (0 ≤ i ≤ 15) is set, at least one error has been detected on sector i+1.
As an example, SEM[2] and SEM[1] bits are set when errors are detected in sector 3 and 2.

FMC BCH interrupt enable register (FMC_BCHIER)

Address offset: 0x250

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EPBRIE:** Decoder Parity Bits Ready Interrupt enable
0: Decoder Parity Bits Ready Interrupt disable
1: Decoder Parity Bits Ready Interrupt enable

Bit 3 **DSRIE:** Decoder Syndrome Ready Interrupt enable
0: Decoder Syndrome Ready Interrupt disable
1: Decoder Syndrome Ready Interrupt enable

Bit 2 **DEFIE:** Decoder Error Found Interrupt enable
0: Decoder Error Found Interrupt disable
1: Decoder Error Found Interrupt enable

Bit 1 **DERIE:** Decoder Error Ready Interrupt enable
0: Decoder Error Ready Interrupt disable
1: Decoder Error Ready Interrupt enable

Bit 0 **DUEIE:** Decoder Uncorrectable Errors Interrupt enable
0: Decoder Uncorrectable Errors Interrupt disable
1: Decoder Uncorrectable Errors Interrupt enable

FMC BCH interrupt and status register (FMC_BCHISR)

Address offset: 0x254

Reset value: 0x0000 0000

This register holds the status of BCH encoder/decoder after processing each sector. When the sequencer is used, this register is automatically cleared and the corresponding interrupt must not be enabled through FMC_BCHIER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRF	DSRF	DEFF	DERF	DUEF
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EPBRF:** Encoder Parity Bits Ready flag
This bit is set when the BCH encoder parity bits are available in FMC_BCHPBR1 to 4.

Bit 3 **DSRF:** Decoder Syndrome Ready flag
This bit is set when the syndrome is ready and DEFF is a valid field.

Bit 2 **DEFF:** Decoder Error Found flag
This bit is set by hardware and cleared by writing 1 to CDEFF.
This bit indicates that at least one error has been detected in the sector.

Bit 1 **DERF:** Decoder Error Ready flag
This bit indicates that the decoder has finished searching errors and identifying their location. The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUEF:** Decoder Uncorrectable Errors flag
This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.

FMC BCH interrupt clear register (FMC_BCHICR)

Address offset: 0x258

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEPBRF	CDSRF	CDEFF	CDERF	CDUEF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CEPBRF:** Clear Encoder Parity Bits Ready flag
Writing 1 clears the EPBRF flag in FMC_BCHISR.

Bit 3 **CDSRF:** Clear Decoder Syndrome Ready flag
Writing 1 clears the DSRF flag in FMC_BCHISR.

Bit 2 **CDEFF:** Clear Decoder Error Found flag
Writing 1 clears the DEFF flag in FMC_BCHISR.

Bit 1 **CDERF:** Clear Decoder Error ready flag
Writing 1 clears the DERF flag in FMC_BCHISR.

Bit 0 **CDUEF:** Clear Decoder Uncorrectable Error flag
Writing 1 clears the DUEF flag in FMC_BCHISR.

FMC BCH parity bits register 1 (FMC_BCHPBR1)

Address offset: 0x260

Reset value: 0x0000 0000

These registers contain the BCH parity bits (BCHPB). For the BCH 4-bit, only BCHPB[51:0] are significant and for the BCH 8-bit BCHPB[103:0] are significant.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[31:0]:** BCH parity bits

FMC BCH parity bits register 2 (FMC_BCHPBR2)

Address offset: 0x264

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[63:32]:** BCH parity bits

FMC BCH parity bits register 3 (FMC_BCHPBR3)

Address offset: 0x268

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[95:64]:** BCH parity bits

FMC BCH parity bits register 4 (FMC_BCHPBR4)

Address offset: 0x26C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BCHPB[103:96]:** BCH parity bits

FMC BCH decoder status register 0 (FMC_BCHDSR0)

Address offset: 0x27C

Reset value: 0x0000 0000

This register contains some fields already available in other registers but that require to be saved when error correction is performed on several sectors at a time (for example a whole NAND flash page). This allows a DMA channel to transfer the content of FMC_BCHDSR0 to 4 to a decoding status buffer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN[3:0]				Res.	Res.	DEF	DUE
								r	r	r	r			r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DEN[3:0]:** Decoder error number

When DEF bit is set, this field indicates the number of errors detected by the BCH decoder.
Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and DUE bit is set.
If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DEN is not significant.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DEF:** Decoder error found

This field indicates that the decoder has finished searching errors and identifying their location.
The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUE:** Decoder uncorrectable error

This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.
If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DUE is not significant.

FMC BCH decoder status register for memory region 1 (FMC_BCHDSR1)

Address offset: 0x280

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 1st and 2nd error bits in EBP1 and EPB2 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are “don’t care”.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP2[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP1[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP2[12:0]**: Error bit position for error number 2

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP1[12:0]**: Error bit position for error number 1

FMC BCH decoder status register for memory region 2 (FMC_BCHDSR2)

Address offset: 0x284

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 3rd and 4th error bits in EBP3 and EPB4 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don’t care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP4[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP3[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP4[12:0]**: Error bit position for error number 4

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP3[12:0]**: Error bit position for error number 3

FMC BCH decoder status register for memory region 3 (FMC_BCHDSR3)

Address offset: 0x288

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 5th and 6th error bits in EBP5 and EPB6 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don't care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP6[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP5[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP6[12:0]**: Error bit position for error number 6

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP5[12:0]**: Error bit position for error number 5

FMC BCH decoder status register for memory region 4 (FMC_BCHDSR4)

Address offset: 0x28C

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 7th and 8th error bits in EBP7 and EPB8 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don't care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP8[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP7[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP8[12:0]**: Error bit position for error number 8

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP7[12:0]**: Error bit position for error number 7

32.12 FMC register map

Table 261. FMC register map

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	1	0	1	1
0x08	FMC_BCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x10	FMC_BCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x18	FMC_BCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x04	FMC_BTR1	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0].				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	FMC_BTR2	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	FMC_BTR3	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	FMC_BTR4	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x20	FMC_CFGR	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	CCLKEN	CLKDIV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0						0	0				0	0	0	0	0																
0x80	FMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR[3:0]			TCLR[3:0]				ECCALG	Res.	ECCEN	PWID		Res.	PBKEN	PWAITEN	Res.
	Reset value							0	0					0	1	1	1	1	1	1	1	1	1	1	0		0	0	0		0	0	
0x84	FMC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWRF	Res.	PEF	Res.	Res.	ISOST	
	Reset value																										1		1			1	1
0x88	FMC_PMEM	MEMHIZ								MEMHOLD								MEMWAIT								MEMSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0

Table 261. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8C	FMC_PATT	ATTHIZ								ATTHOLD								ATTWAIT								ATTSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0
0x90	FMC_HPR	HPR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	FMC_HECCR	HECC[[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	Reserved	Reserved																															
0x104	FMC_BWTR1	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	FMC_BWTR2	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FMC_BWTR3	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	FMC_BWTR4	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x180	FMC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEE	IHLE	IREE
	Reset value																														0	0	0
0x184	FMC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEF	IHLF	IREF
	Reset value																														0	0	0
0x188	FMC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIFEF	CIHLF	CIREF
	Reset value																														0	0	0
0x200	FMC_CSQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSQSTART
	Reset value																																0
0x204	FMC_ CSQCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2								CMD1								ACYNBR			Res.	Res.	DMADEN	CMD2EN	Res.
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	
0x208	FMC_ CSQCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2								RCMD1								Res.	Res.	Res.	Res.	Res.	DMADEN	RCMD2EN	SQSDTEN
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x20C	FMC_ CSQCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T	Res.	Res.	SNBR						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0			0	0	0	0	0	0								
0x210	FMC_CSQAR1	ADDCA4								ADDC3								ADDC2								ADDC1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 261. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x214	FMC_CSQAR2	SAO																Res.	Res.	Res.	Res.	NANDCEN1	NANDCEN0	Res.	Res.	ADDC5							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0					0	0			0	0	0	0	0	0	0	0
0x220	FMC_CSQIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCIE	SUEIE	SEIE	SCIE	TCIE
	Reset value																												0	0	0	0	0
0x224	FMC_CSQISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCF	SUEF	SEF	SCF	TCF
	Reset value																												0	0	0	0	0
0x228	FMC_CSQICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDTCF	CSUEF	CSEF	CSCF	CTCF
	Reset value																												0	0	0	0	0
0x22C	Reserved	Reserved																															
0x230	FMC_CSQEMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x240	Reserved	Reserved																															
0x250	FMC_BCHIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
	Reset value																												0	0	0	0	0
0x254	FMC_BCHISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRF	DSRF	DEFF	DERF	DUEF
	Reset value																												0	0	0	0	0
0x258	FMC_BCHICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEPBRF	CDSRF	CDEFF	CDERF	CDUEF
	Reset value																												0	0	0	0	0
0x260	FMC_BCHPBR1	BCHPB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x264	FMC_BCHPBR2	BCHPB[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x268	FMC_BCHPBR3	BCHPB[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x26C	FMC_BCHPBR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
	Reset value																									0	0	0	0	0	0	0	0
0x27C	FMC_BCHDSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN					Res.	Res.	DEF	DUE
	Reset value																									0	0	0	0			0	0
0x280	FMC_BCHDSR1	Res.	Res.	Res.	EBP2[12:0]													Res.	Res.	Res.	EBP1[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x284	FMC_BCHDSR2	Res.	Res.	Res.	EBP4[12:0]													Res.	Res.	Res.	EBP3[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x288	FMC_BCHDSR3	Res.	Res.	Res.	EBP6[12:0]													Res.	Res.	Res.	EBP5[12:0]												
	Reset value																																
0x28C	FMC_BCHDSR4	Res.	Res.	Res.	EBP8[12:0]													Res.	Res.	Res.	EBP7[12:0]												
	Reset value																																

Table 261. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	FMC_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																											0	0	0	0	0	0
0x304	FMC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																											0	0	0	0	0	0
0x308	FMC_CIDCFG_ LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																											0	0	0	0	0	0
0x30C	FMC_CIDCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC6	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x310	FMC_SEMCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x314	FMC_CIDCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x318	FMC_SEMCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x31C	FMC_CIDCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x320	FMC_SEMCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x324	FMC_CIDCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x328	FMC_SEMCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x32C	FMC_CIDCFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0

Table 261. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x330	FMC_SEMCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x334	FMC_CIDCFGR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMWLC7	SEMWLC6	SEMWLC5	SEMWLC4	SEMWLC3	SEMWLC2	SEMWLC1	SEMWLC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEMEN	CFEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0
0x338	FMC_SEMCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEMCID[2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x3E8	FMC_HWCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID_WIDTH[3:0]				MAX_CID[3:0]			
	Reset value	0x0000 0037																															
0x3EC	FMC_HWCFGR2	Res.	Res.	Res.	SECURE	Res.	Res.	Res.	PRIVILEGE	SDRAM2_BASE[3:0]				SDRAM1_BASE[3:0]				NAND_BASE[3:0]				SDRAM_RBASE[3:0]				NOR_BASE[3:0]				RD_LN2PTH[3:0]			
	Reset value	0x11DC 4772																															
0x3F0	FMC_HWCFGR1	RA_LN2DPTH[3:0]				WR_LN2DPTH[3:0]				WD_LN2DPTH[3:0]				WA_LN2DPTH[3:0]				ID_SIZE[3:0]				Res.	Res.	Res.	SDRAM_SEL	Res.	Res.	Res.	NAND_ECC	Res.	Res.	Res.	NAND_SEL
	Reset value	0x2232 9011																															
0x3F4	FMC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0020																															
0x3F8	FMC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	FMC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

33 Octo-SPI interface (OCTOSPI)

33.1 OCTOSPI introduction

The OCTOSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the OCTOSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The OCTOSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

33.2 OCTOSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, and octal communication
- Dual memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AHB interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.
- Dual chip select support

33.3 OCTOSPI implementation

Table 262. OCTOSPI implementation

OCTOSPI feature	OCTOSPI1
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Asynchronous AHB clock versus kernel clock	X
Dual AHB interface	X
Functional modes: indirect, automatic status-polling, and memory-mapped	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8, 16, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Bus error bit	-
Address offset	-
Compliant with dual-OCTOSPI arbiter (communication regulation)	-
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
High-speed interface	-
LP/HPDMA interface	X
Dual chip select	X
Extended external memory	-
Prefetch disable	-
Prefetch hardware software	-

33.4 OCTOSPI functional description

33.4.1 OCTOSPI block diagram

Figure 198. OCTOSPI block diagram in octal configuration

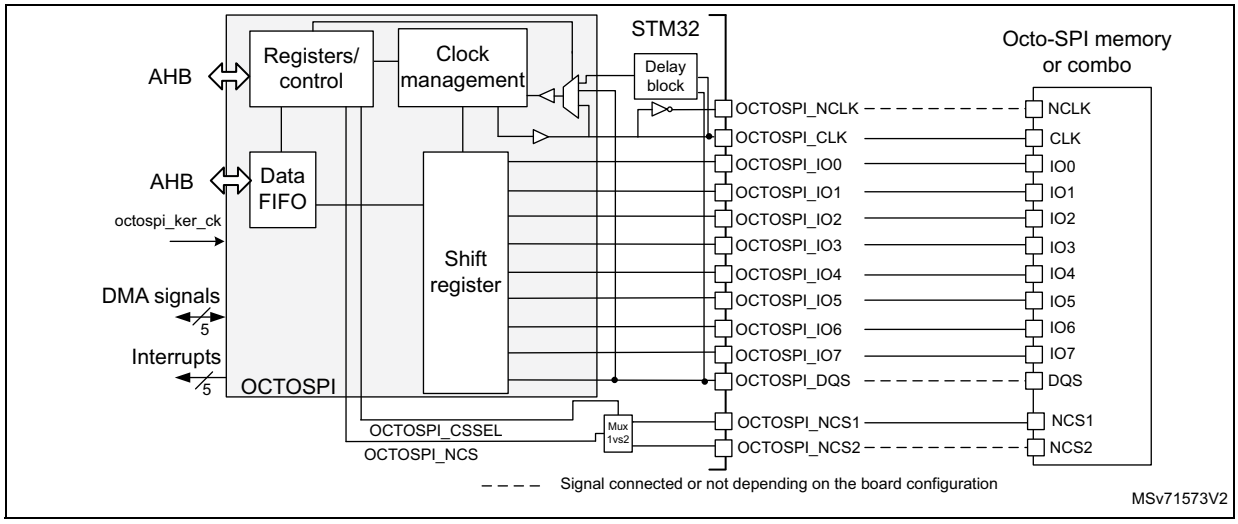


Figure 199. OCTOSPI block diagram in quad configuration

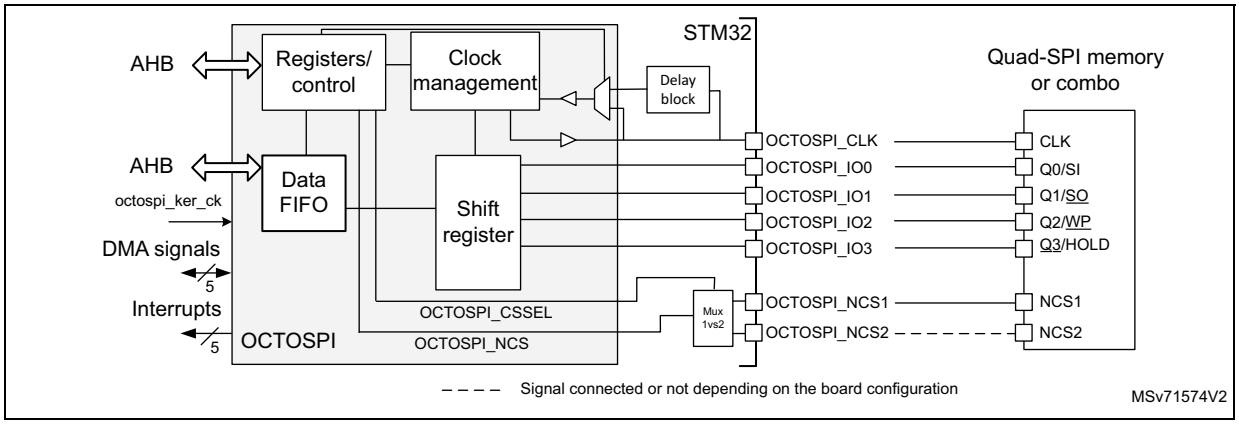
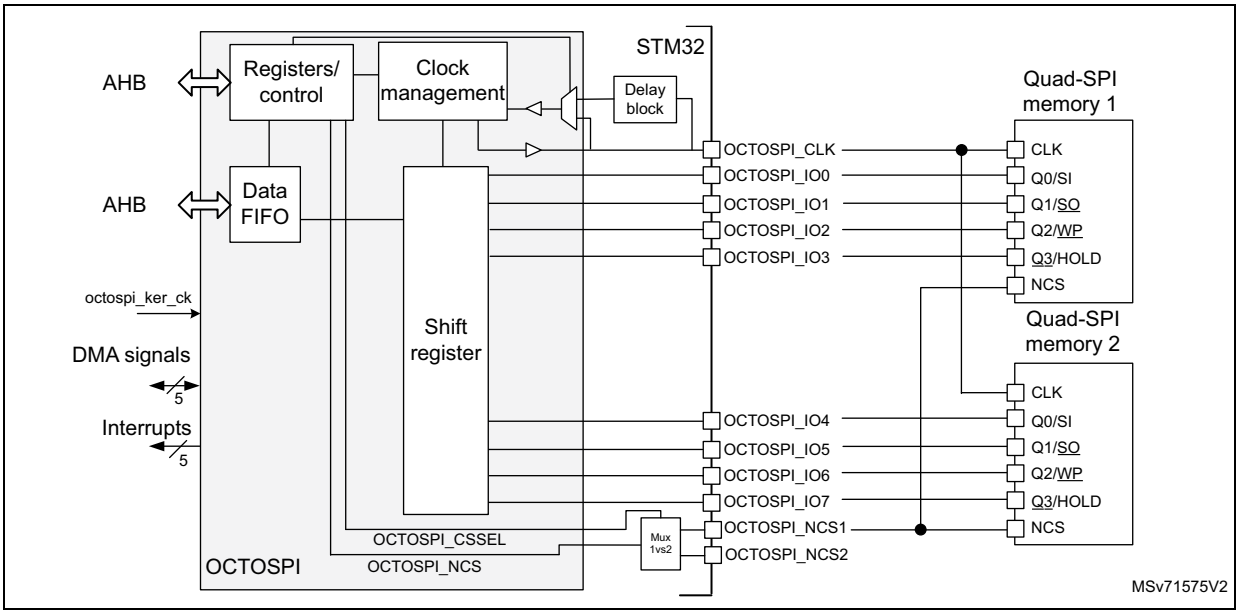


Figure 200. OCTOSPI block diagram in dual-quad configuration



33.4.2 OCTOSPI pins and internal signals

Table 263. OCTOSPI input/output pins

Pin name	Type	Description
OCTOSPI_NCLK	Output	OCTOSPI inverted clock to support 1.8 V HyperBus protocol. The inversion must be activated only in this case, by setting the INVCLK bit in *Section 22.3.14: Advanced I/O configurations*.
OCTOSPI_CLK		OCTOSPI clock
OCTOSPI_IOn (n = 0 to 7)	Input/output	OCTOSPI data pins
OCTOSPI_NCS1,2	Output	Chip select for the memory
OCTOSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all OCTOSPI input/output pins to avoid any data corruption.

Table 264. OCTOSPI internal signals

Signal name	Type	Description
octospi_hclk	Input	OCTOSPI AHB clock
octospi_ker_ck	Input	OCTOSPI kernel clock
octospi_dma	N/A	DMA request signal
octospi_it	Output	Global interrupt line (see *Table 268* for the multiple sources of interrupt)

a. When applicable.

33.4.3 OCTOSPI interface to memory modes

The OCTOSPI supports the following protocols:

- regular-command protocol
- HyperBus protocol

The OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time)
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

33.4.4 OCTOSPI regular-command protocol

When in regular-command protocol, the OCTOSPI communicates with the external device using commands. Each command can include the following phases:

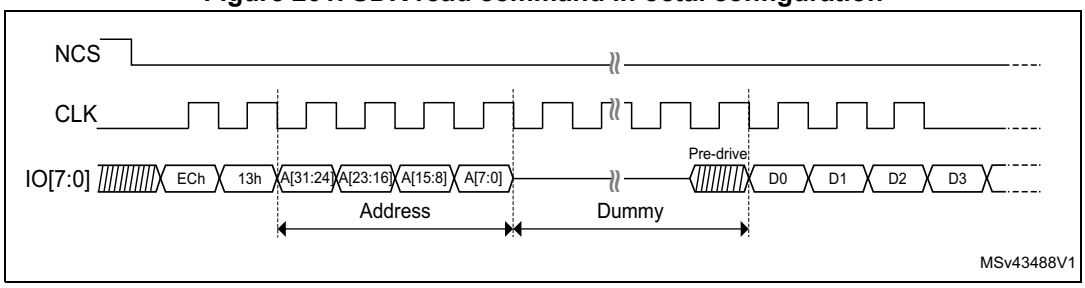
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 201. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in OCTOSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of OCTOSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of OCTOSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in OCTOSPI_WIR, and the instruction format is specified in OCTOSPI_WCCR. The instruction used for the read operation and the instruction format are specified in OCTOSPI_IR and OCTOSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in OCTOSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in OCTOSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in OCTOSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in OCTOSPI_WCCR. The address format for the read operation is specified in OCTOSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in OCTOSPI_CCR. The bytes to be sent are specified in OCTOSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in OCTOSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in OCTOSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in OCTOSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in OCTOSPI_WABR, and the alternate byte format is specified in OCTOSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in OCTOSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-SPI, the quad-SPI, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in OCTOSPI_WTCR. The dummy cycles for the read operation are specified in OCTOSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in OCTOSPI_DLR. In this mode, the data to be sent to the external device must be written to OCTOSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading OCTOSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in OCTOSPI_DLR, and the data received from the external device can be obtained by reading OCTOSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in OCTOSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

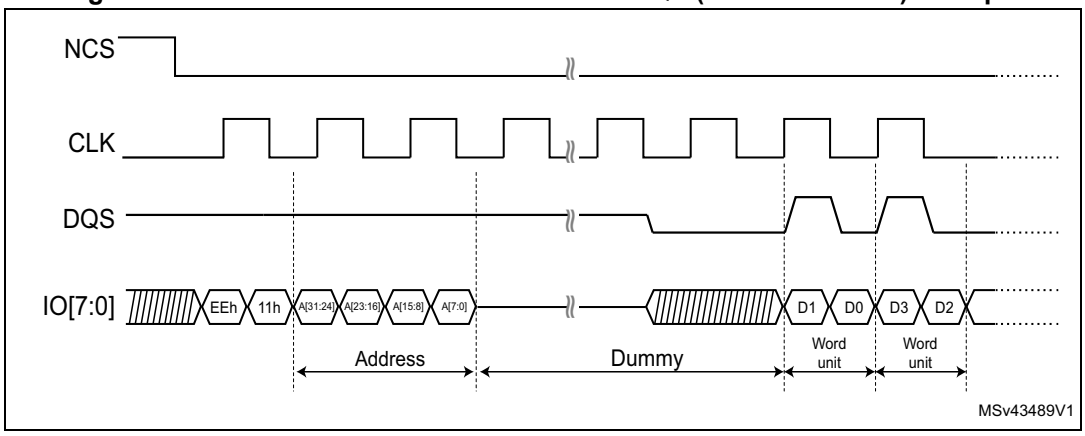
In memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in OCTOSPI_CCR.

Figure 202. DTR read in octal-SPI mode with DQS (Macronix mode) example



33.4.5 OCTOSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the OCTOSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

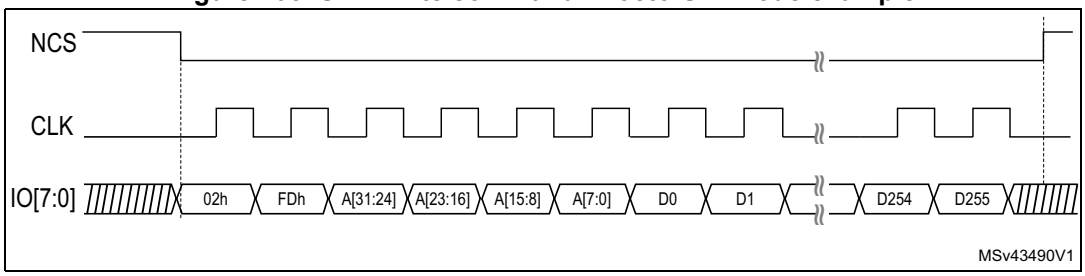
Single-data rate (SDR) mode

By default, all the phases operate in SDR mode.

In this mode, when the OCTOSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 203. SDR write command in octo-SPI mode example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in OCTOSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in OCTOSPI_WCCR. The DTR mode for each phase of the read operations is specified in OCTOSPI_CCR.

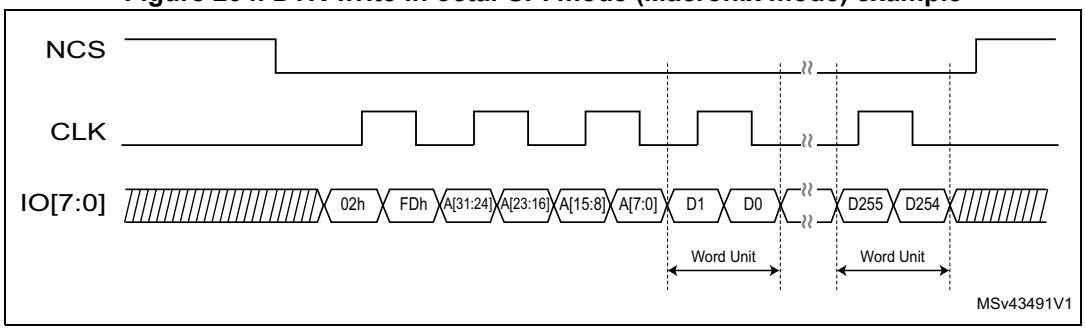
In DTR mode, when the OCTOSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 204. DTR write in octal-SPI mode (Macronix mode) example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Dual-quad configuration

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 011, two external Quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in OCTOSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in OCTOSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

33.4.6 HyperBus protocol

The OCTOSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

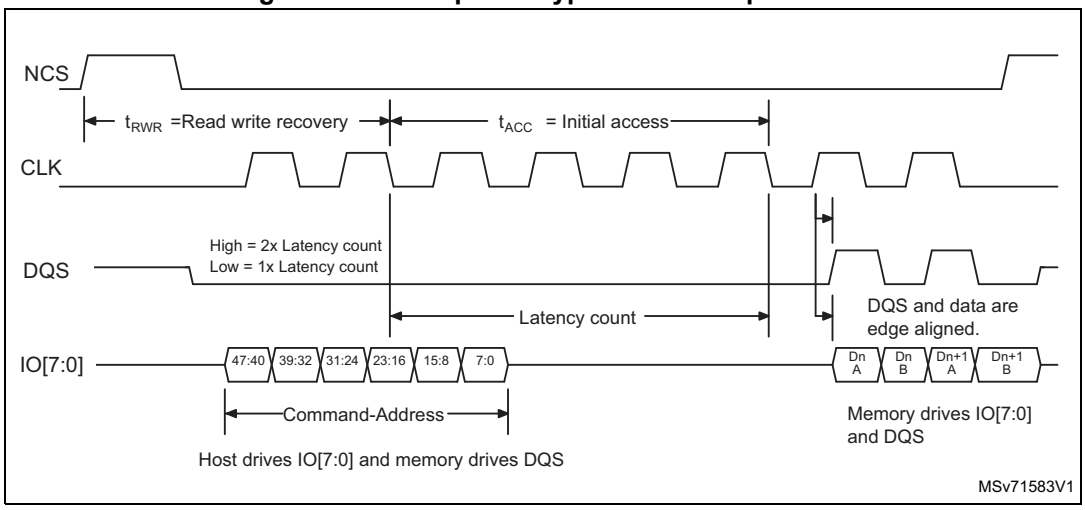
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 205. Example of HyperBus read operation



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 265. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in OCTOSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of OCTOSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in OCTOSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in OCTOSPI_HLCR) according to the memory latency

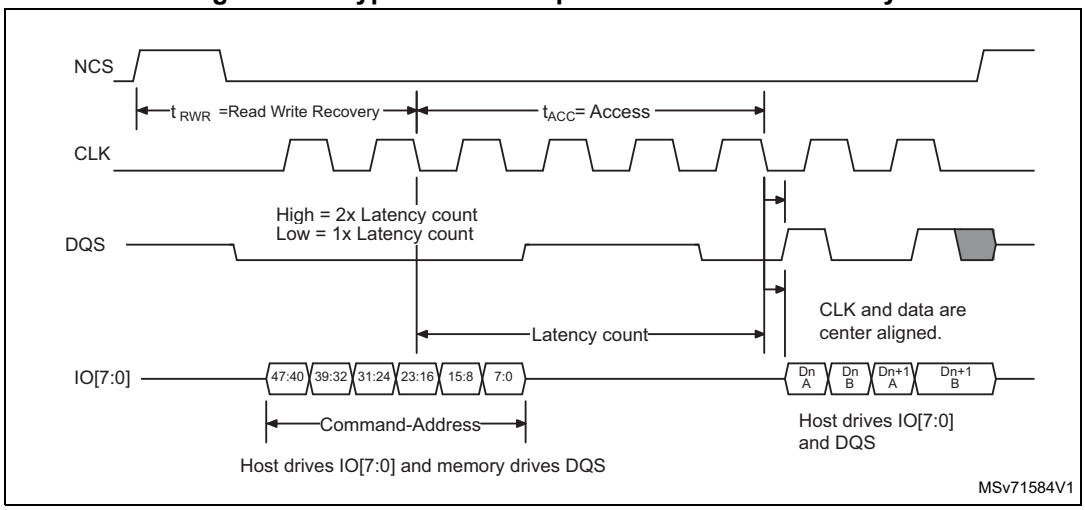
During the read operation, the DQS is used by the device, in two ways (see *Figure 205*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the OCTOSPI, during the data phase, for write data masking.

Figure 206. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the OCTOSPI to meet the device request.

Figure 207. HyperBus read operation with additional latency

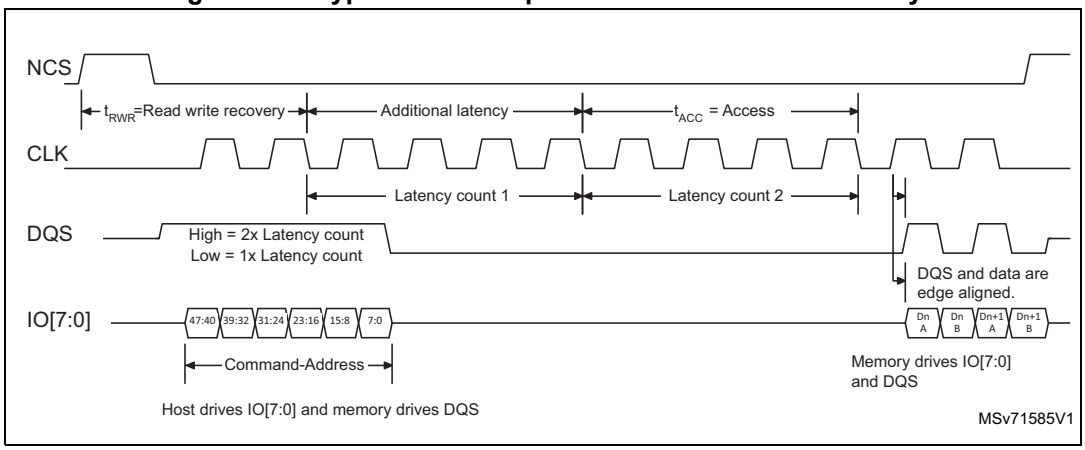
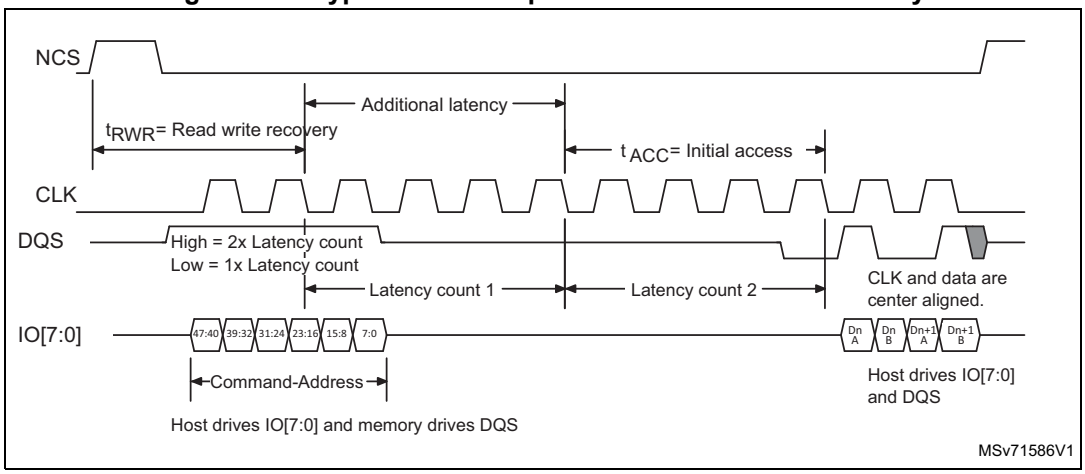


Figure 208. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

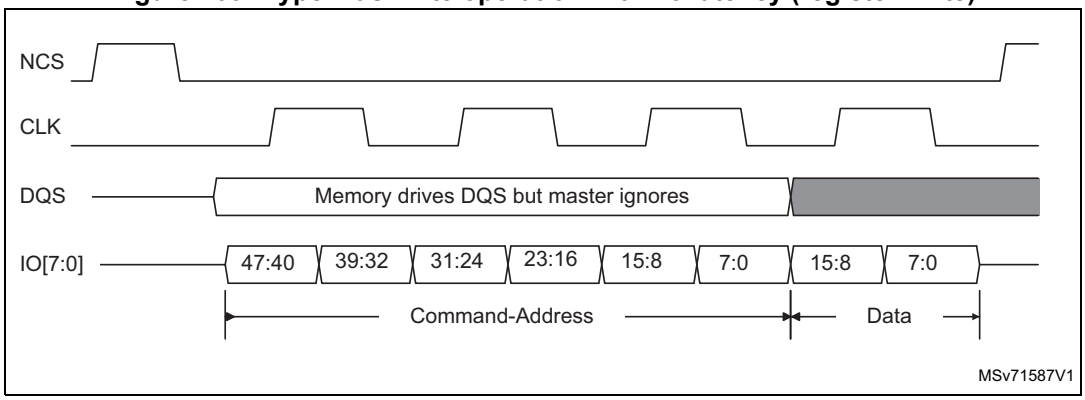
The latency can be forced to 2 x t_{ACC} by setting LM in OCTOSPI_HLCR.

In this OCTOSPI latency mode, the state of the DQS signal is not taken into account by the OCTOSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in OCTOSPI_HLCR.

Figure 209. HyperBus write operation with no latency (register write)

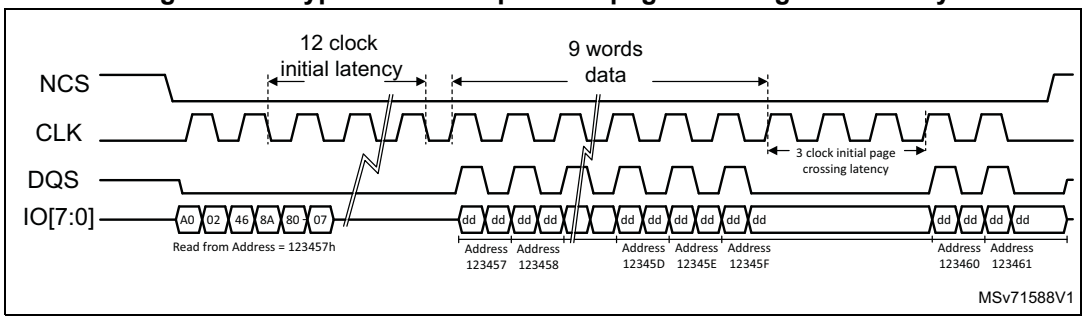


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 210. HyperBus read operation page crossing with latency



33.4.7 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The OCTOSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in OCTOSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated OCTOSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (OCTOSPI_CCR, OCTOSPI_TCR, OCTOSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the OCTOSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in OCTOSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Restarting after an interrupted transfer

When a read or write operation is interrupted by a timeout or communication regulation feature, the Octo-SPI interface, as soon as possible after getting back the port ownership, reissues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

33.4.8 OCTOSPI operating mode introduction

The OCTOSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

33.4.9 OCTOSPI indirect mode

In indirect mode, the commands are started by writing to the OCTOSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in OCTOSPI_CR, the OCTOSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading OCTOSPI_DR.

In indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The OCTOSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as

defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.

- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in OCTOSPI_SR, and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the OCTOSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (OCTOSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (OCTOSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (OCTOSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode passes through a 64-byte FIFO that is internal to the OCTOSPI. FLEVEL in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in OCTOSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in OCTOSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

33.4.10 OCTOSPI automatic status-polling mode

In automatic status-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in OCTOSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in OCTOSPI_CR, the AND-match mode is activated: SMF is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in OCTOSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the OCTOSPI is disabled (EN = 0).

OCTOSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in OCTOSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. The memory must then be configured in fixed latency.

33.4.11 OCTOSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the OCTOSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be

programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.

- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

The controller can apply an automatic address offset to simplify memory area code switching, such as during a firmware update (if the address offset feature is available in the product implementation table). To enable this feature, set the ADOFFEN bit in the XSPI_CR register and configure the offset value in the ADOFF[4:0] bitfield of the XSPI_DCR1 register. The offset is modulo 256 bytes.

33.4.12 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

33.4.13 OCTOSPI system configuration

The OCTOSPI is configured using OCTOSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.

The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and OCTOSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

33.4.14 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

Number of bytes in the device = $2^{DEVSIZE+1}$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in OCTOSPI_HLCR.

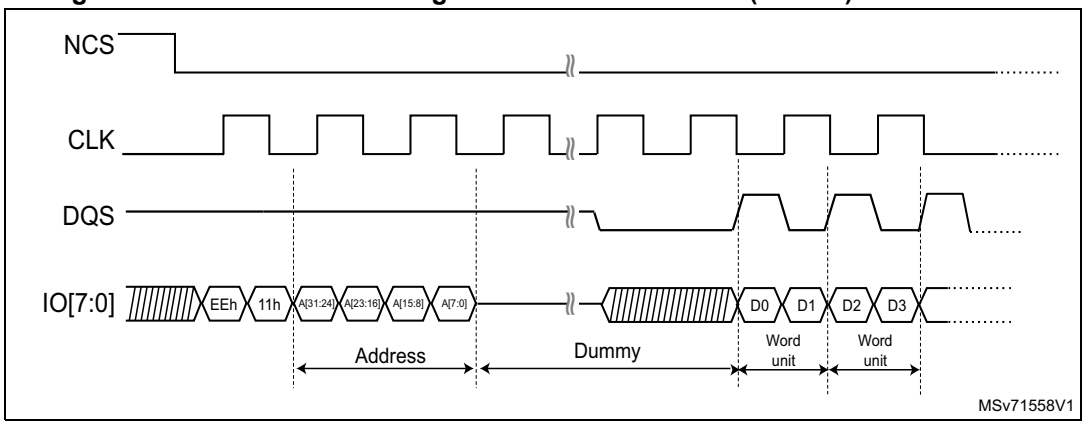
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the OCTOSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 211. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 202: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 266: OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 212. OctaRAM read operation with reverse data ordering D1/D0

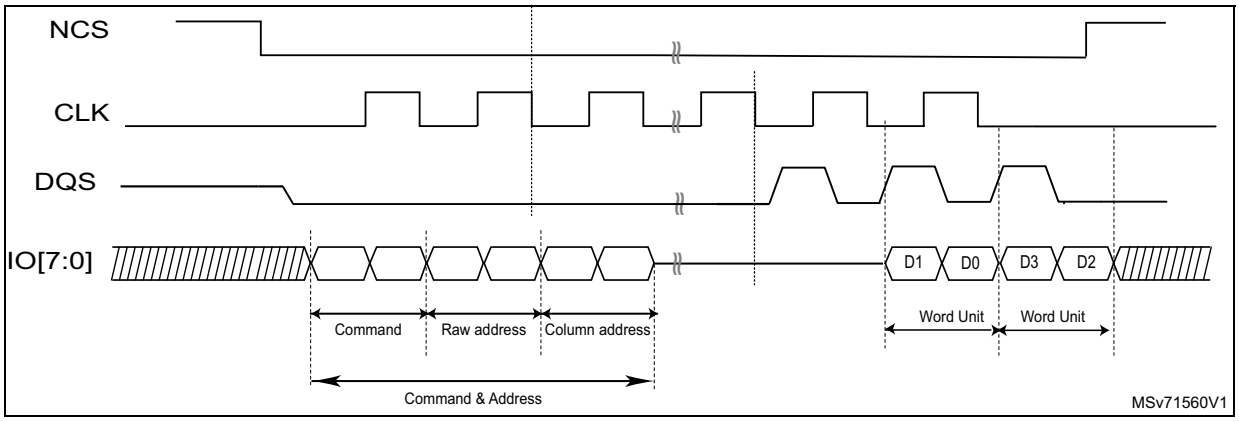


Table 266. OctaRAM command address bit assignment (based on 64-Mbyte(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64-Mbyte OctaRAM address assignment:
Row Address [RA12:RA0]: 8K. Column address [CA9:CA0]: 1K. 64-Mbyte density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.
- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any others configurations described in this section. MTYP[2:0] = 010 for this standard mode.

33.4.15 OCTOSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.

7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

When the OCTOSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.
7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

OCTOSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

OCTOSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay.

This shift is performed by a delay hardware element located outside of the controller into the RCC (RCC_DLYCFGR) or implemented on an external dedicated block delay peripheral

depending on the product (refer to the product reference manual for information about how to manage and configure the delay).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in OCTOSPI_DCR1.

33.4.16 OCTOSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the targeted address in OCTOSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

33.4.17 OCTOSPI error management

The following errors set the TEF flag in OCTOSPI_SR and generates an interrupt if enabled (TEIE = 1 in OCTOSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]).

- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the OCTOSPI generates an AHB slave error in the following situations, setting the BERRF flag in XSPI_SR and generating an interrupt if enabled (BERRIE = 1 in XSPI_CR) (if the Bus error bit feature is available in the product implementation table, otherwise the Slave bus error response is generated at system level as transaction response with no dedicated bit into the controller status register):

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read or write address (including the address offset if ADOFFEN bit is set) exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing.
- The OCTOSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in OCTOSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AHB bus transfer size (HSIZE) is different from 4 bytes (only for memory-mapped mode).
- Bit DMM is set in OCTOSPI_CR (dual-memory configuration) and octal mode is set.

33.4.18 OCTOSPI BUSY and ABORT

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in OCTOSPI_DR, it is recommended to disable the DMA channel before aborting the OCTOSPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

33.4.19 OCTOSPI reconfiguration or deactivation

Before any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

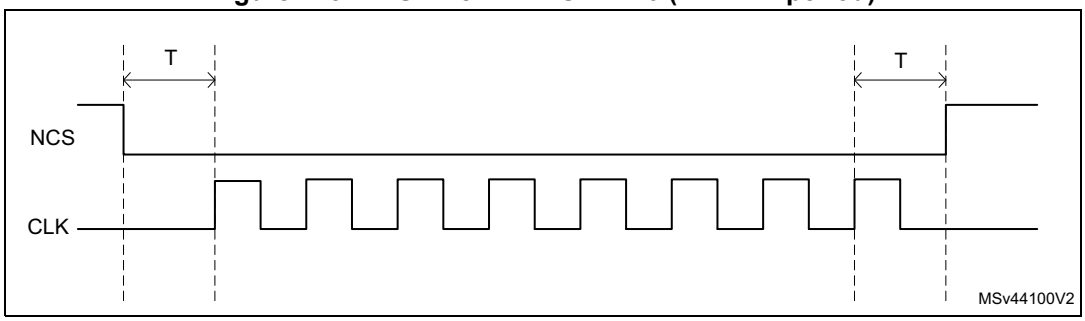
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

33.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

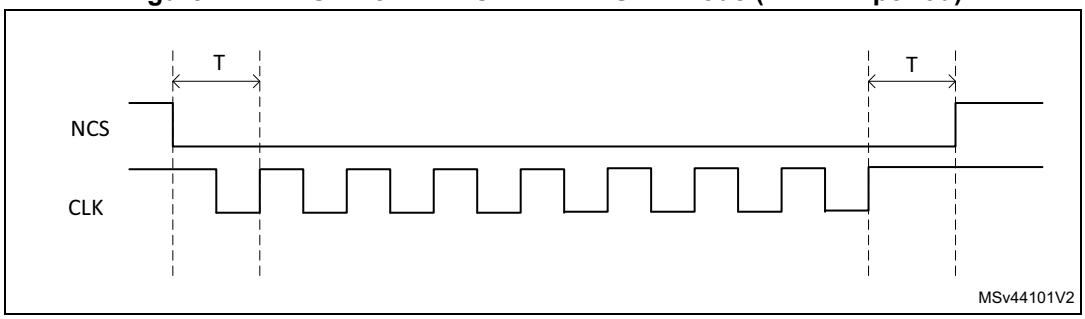
When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 213. NCS when CKMODE = 0 (T = CLK period)



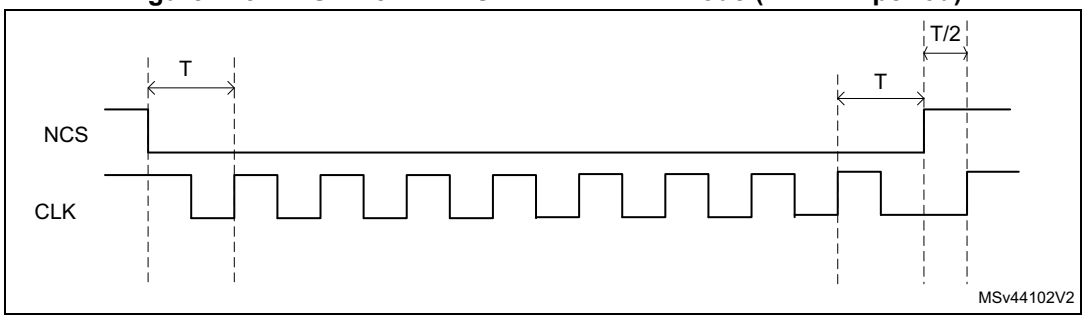
When CKMODE = 1 (clock mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 214. NCS when CKMODE = 1 in SDR mode (T = CLK period)



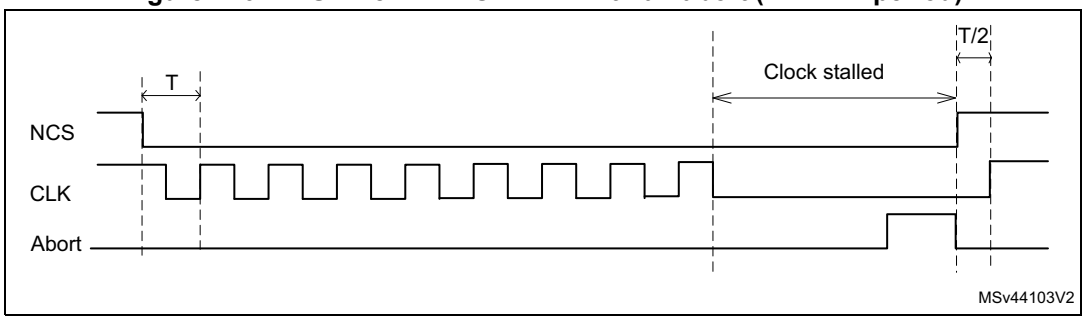
When the CKMODE = 1 (clock mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 215. NCS when CKMODE = 1 in DTR mode (T = CLK period)



When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later (see the figure below).

Figure 216. NCS when CKMODE = 1 with an abort (T = CLK period)

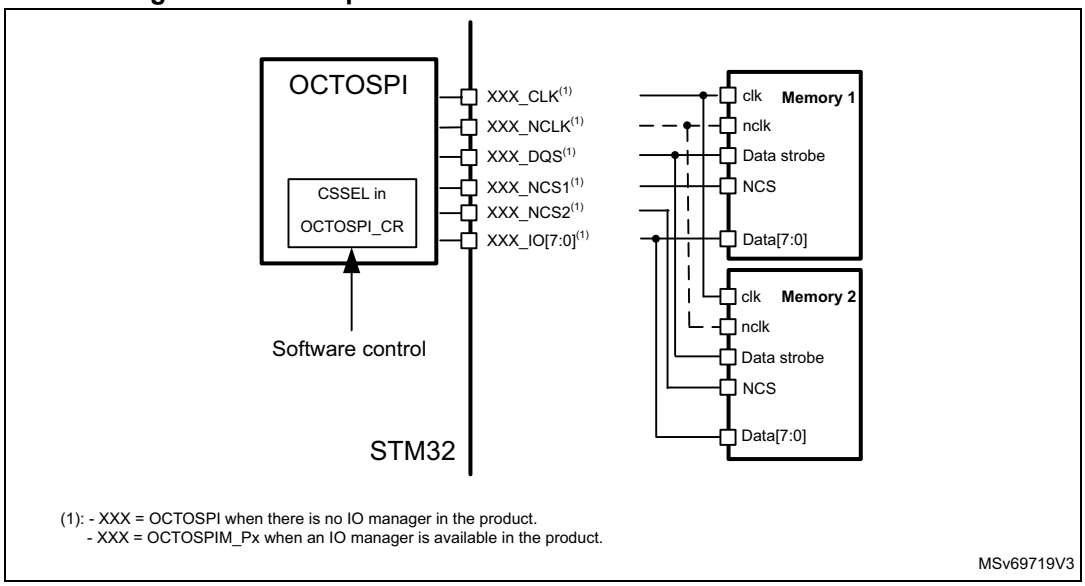


In dual memory configuration, the NCS2 behaves exactly the same as the NCS1.
The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

33.4.21 Software control of two external memories

One OCTOSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the OCTOSPI. The OCTOSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in OCTOSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus. From a memory-map point of view, these two memories are located in the same address.

Figure 217. Example of software control of two external memories



33.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 267. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.

7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

33.6 OCTOSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 268. OCTOSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

33.7 OCTOSPI registers

33.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MSEL	FMODE[1:0]		Res.	Res.	Res.	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FTHRES[5:0]						Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
		rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Note: This bit is mirrored in bit 7. This bit can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the OCTOSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode (relevant in regular-command protocol only)

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **CSSEL**: Chip-select selection

This bit indicates to OCTOSPI I/O manager if it must copy NCS to NCS1 or NCS2.

0: NCS I/O manager input is copied to NCS1 I/O manager output.

1: NCS I/O manager input is copied to NCS2 I/O manager output.

Note: This bit can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used to determine a match during the automatic status-polling mode.

0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.

1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop

This bit determines if the automatic status-polling mode is stopped after a match.

0: Automatic status-polling mode is stopped only by abort or by disabling the OCTOSPI.

1: Automatic status-polling mode stops as soon as there is a match.

Note: This bit can be modified only when BUSY = 0.

Bit 21 **BERRIE**: Bus error interrupt enable

This bit enables the bus error interrupt (if Bus error bit feature is available in the product implementation table).

0: Interrupt disabled

1: Interrupt enabled

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable

This bit enables the status-match interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **FTHRES[5:0]**: FIFO threshold level
This fbitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.
000000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.
000001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.
...
111111: FTF is set if there are 64 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 64 valid bytes can be read from the FIFO in indirect-read mode.
Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[5:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration
This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity
0: Dual-memory configuration disabled
1: Dual-memory configuration enabled
Note: This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ADOFFEN**: Address offset enable
An offset may be applied on the memory address. The offset value is set into ADOFF[4:0].
0: Address offset disabled
1: Address offset enabled
Note: This bit can be modified only when BUSY = 0. This bit is present only if the address offset feature is available in the product implementation table.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the OCTOSPI.

0: OCTOSPI disabled

1: OCTOSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

33.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1)

Address offset: 0x0008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADOFF[4:0]					MTYP[2:0]			Res.	Res.	Res.	DEVSIZE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLY BYP	Res.	FRCK	CKMODE
		rw	rw	rw	rw	rw	rw					rw		rw	rw

Bits 31:27 **ADOFF[4:0]**: Address offset

This bitfield defines the memory address offset value added to the current address requested to be sent by the controller to hit the external memory using the following formula:
Offset value = $2^{[ADOFF+8]}$. (modulo 256 bytes)
Maximum allowed value is 19 since the memory-mapped mode is limited to 256 Mbytes address space. For greater value, the address offset will not be applied to the address sent by the controller.
This bitfield is relevant only if ADOFFEN is set and if FMODE[1:0] = 11 (memory-mapped mode) and if the address offset feature is available in the product implementation table.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.
000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.
Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.
001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.
010: Standard mode
011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).
100: HyperBus memory mode, the protocol follows the HyperBus specification.
101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.
Others: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.
0x0: NCS stays high for at least 1 cycle between external device commands.
0x1: NCS stays high for at least 2 cycles between external device commands.
...
0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0/mode 3

This bit indicates the level taken by the CLK between commands (when NCS = 1).

0: CLK must stay low while NCS is high (chip-select released). This is referred to as clock mode 0.

1: CLK must stay high while NCS is high (chip-select released). This is referred to as clock mode 3.

33.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the OCTOSPI1_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

33.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

33.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFRESH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Others: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the divided clock period: if OCTOSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

33.7.6 OCTOSPI status register (OCTOSPI_SR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FLEVEL[6:0]							Res.	BERRF	BUSY	TOF	SMF	FTF	TCF	TEF
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **FLEVEL[6:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 64 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **BERRF**: Bus error flag

This bit is set when a bus error occurs. It is cleared by writing 1 to CBERRF. This bit has a meaning if Bus error bit feature is available in the product implementation table.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag
This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag
This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_PSMAR).
It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag
In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.
It is cleared automatically as soon as the threshold condition is no longer true.
In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag
This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag
This bit is set in indirect mode when an invalid address is being accessed in indirect mode.
It is cleared by writing 1 to CTEF.

33.7.7 OCTOSPI flag clear register (OCTOSPI_FCR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	CSMF	Res.	CTCF	CTEF
									w		w	w		w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CBERRF**: Clear bus error flag
Writing 1 clears the BERRF flag in the XSPI_SR register (if Bus error bit feature is available in the product implementation table).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the OCTOSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the OCTOSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the OCTOSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the OCTOSPI_SR register.

33.7.8 OCTOSPI data length register (OCTOSPI_DLR)

Address offset: 0x0040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length
Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.
All 1's in indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSIZE.
0x0000_0000: 1 byte is to be transferred.
0x0000_0001: 2 bytes are to be transferred.
0x0000_0002: 3 bytes are to be transferred.
0x0000_0003: 4 bytes are to be transferred.
...
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.
0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.
DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.
This bitfield has no effect in memory-mapped mode.

33.7.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In HyperBus protocol, this field must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

33.7.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes. Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

33.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

33.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

Address offset: 0x0088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match

33.7.13 OCTOSPI polling interval register (OCTOSPI_PIR)

Address offset: 0x0090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between a read during the automatic status-polling phases

33.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

Address offset: 0x0100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate

This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode

This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size

This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate

This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

33.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

33.7.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

33.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address.

33.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period
After each access in memory-mapped mode, the OCTOSPI prefetches the subsequent bytes and hold them in the FIFO.
This bitfield indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

33.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Others: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

33.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

33.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

33.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

33.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

33.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

33.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction

Instruction to be sent to the external SPI device

33.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

33.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.
0: Latency on write accesses
1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.
0: Variable initial latency
1: Fixed latency

33.7.28 OCTOSPI hardware configuration register 2 (OCTOSPI_HWCFGR2)

Address offset: 0x03EC

Reset value: 0x0000 0X01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSINT[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM16[3:0]				ARBM[3:0]				RDFT[3:0]				FRCK[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **HSINT[3:0]**: High-speed interface
0000: Standard interface
0001: High-speed interface
Others: Reserved

Bits 15:12 **MEM16[3:0]**: Size of the SPI memory supported
0000: 8-bit SPI memory supported
0001: 16-bit SPI memory supported
Others: Reserved

Bits 11:8 **ARBM[3:0]**: Arbitration configuration
0000: No associated OCTOSPI I/O manager
0001: This instance is OCTOSPI1.
0010: This instance is OCTOSPI2.
0011: This instance is OCTOSPI3.
Others: Reserved

Bits 7:4 **RDFT[3:0]**: Reduced features
0000: All features available
0001: Only some feature are supported (see *Section 33.3: OCTOSPI implementation*)
Others: Reserved

Bits 3:0 **FRCK[3:0]**: Free running clock
0000: The delay block does not require a free running clock.
0001: The DLL/delay block receives a free running clock from OCTOSPIs.
Others: reserved

33.7.29 OCTOSPI hardware configuration register (OCTOSPI_HWCFGR)

Address offset: 0x03F0

Reset value: 0xX130 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSSRV[3:0]				MMW[3:0]				IDL[3:0]				PRES[7:4]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRES[3:0]				FIFO[7:0]								AXI[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CSSRV[3:0]**: CSSEL reset value
0000: CSSEL reset value is 1 (OCTOSPI2).
0001: CSSEL reset value is 0 (OCTOSPI1 or 3).
Others: Reserved

Bits 27:24 **MMW[3:0]**: Memory-mapped write
OCTOSPI memory-mapped write
0: No memory-mapped write
1: Memory-mapped write present

Bits 23:20 **IDL[3:0]**: ID Length
AXI ID Length

Bits 19:12 **PRES[7:0]**: Prescaler
Prescaler reset value

Bits 11:4 **FIFO[7:0]**: FIFO depth
FIFO size in 32-bit words

Bits 3:0 **AXI[3:0]**: OCTOSPI interface
0000: AHB interface
0001: AXI interface
Others: Reserved

33.7.30 OCTOSPI version register (OCTOSPI_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0053

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VER[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **VER[7:0]**: OCTOSPI version

33.7.31 OCTOSPI identification register (OCTOSPI_IDR)

Address offset: 0x03F8

Reset value: 0x0014 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: OCTOSPI identification

33.7.32 OCTOSPI hardware magic identification register (OCTOSPI_MIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MID[31:0]**: OCTOSPI magic identification

33.7.33 OCTOSPI register map

Table 269. OCTOSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	OCTOSPI_CR	Res.	MSEL	FMDOE[1:0]		Res.	Res.	Res.	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	FTHRES[5:0]						Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
	Reset value		0	0	0				0	0	0	0	0	0	0	0	0			0	0	0	0	0	0		0		0	0	0	0	0
0x0004	Reserved	Reserved																															
0x0008	OCTOSPI_DCR1	ADOFF[4:0]					MTYP [2:0]			Res.	Res.	Res.	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x000C	OCTOSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x0010	OCTOSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x0014	OCTOSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018-0x001C	Reserved	Reserved																															
0x0020	OCTOSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[6:0]							Res.	BERRF	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0024	OCTOSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																										0		0	0		0	0

Table 269. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0028-0x003C	Reserved	Reserved																															
0x0040	OCTOSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	Reserved	Reserved																															
0x0048	OCTOSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	OCTOSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054-0x007C	Reserved	Reserved																															
0x0080	OCTOSPI_ PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	Reserved	Reserved																															
0x0088	OCTOSPI_ PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	Reserved	Reserved																															
0x0090	OCTOSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094-0x00FC	Reserved	Reserved																															
0x0100	OCTOSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0104	Reserved	Reserved																															
0x0108	OCTOSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x010C	Reserved	Reserved																															
0x0110	OCTOSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114-0x011C	Reserved	Reserved																															
0x0120	OCTOSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124-0x012C	Reserved	Reserved																															

Table 269. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0130	OCTOSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134-0x013C	Reserved	Reserved																															
0x0140	OCTOSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]			Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0144	Reserved	Reserved																															
0x0148	OCTOSPI_WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x014C	Reserved	Reserved																															
0x0150	OCTOSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0154-0x015C	Reserved	Reserved																															
0x0160	OCTOSPI_WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0164-0x017C	Reserved	Reserved																															
0x0180	OCTOSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]			Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0184	Reserved	Reserved																															
0x0188	OCTOSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x018C	Reserved	Reserved																															
0x0190	OCTOSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0194-0x019C	Reserved	Reserved																															
0x01A0	OCTOSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01A4-0x01FC	Reserved	Reserved																															
0x0200	OCTOSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0
0x022C-0x03E8	Reserved	Reserved																															

Table 269. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03EC	OCTOSPI_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSINT[3:0]				MEM16[3:0]				ARBM[3:0]				RDFT[3:0]				FRCK[3:0]			
	Reset value	0x0000 0X01																															
0x03F0	OCTOSPI_ HWCFGR	CSSRV[3:0]				MMW[3:0]				IDL[3:0]				PRES[7:0]								FIFO[7:0]								AXI[3:0]			
	Reset value	0xX130 0100																															
0x03F4	OCTOSPI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VER[7:0]							
	Reset value	0x0000 0053																															
0x03F8	OCTOSPI_IDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x03FC	OCTOSPI_MIDR	MID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

34 Secure digital input/output MultiMediaCard interface (SDMMC)

34.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

34.2 SDMMC implementation

Table 270. SDMMC features

SDMMC modes/features[1]	SDMMC1	SDMMC2	SDMMC3
Variable delay	X	X	X
SDMMC_CKIN	X	X	X
SDMMC_CDIR, SDMMC_D0DIR	X	X	X
SDMMC_D123DIR	X	X	X
D7:D4	X	X	-

1. X = supported.

34.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 218. SDMMC "no response" and "no data" operations

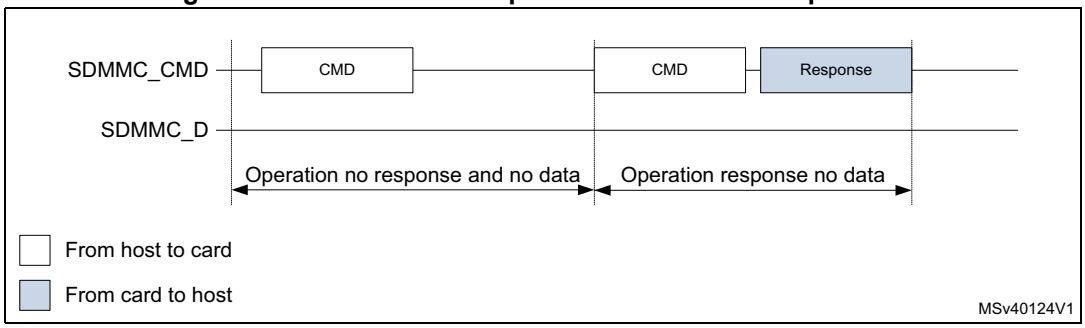
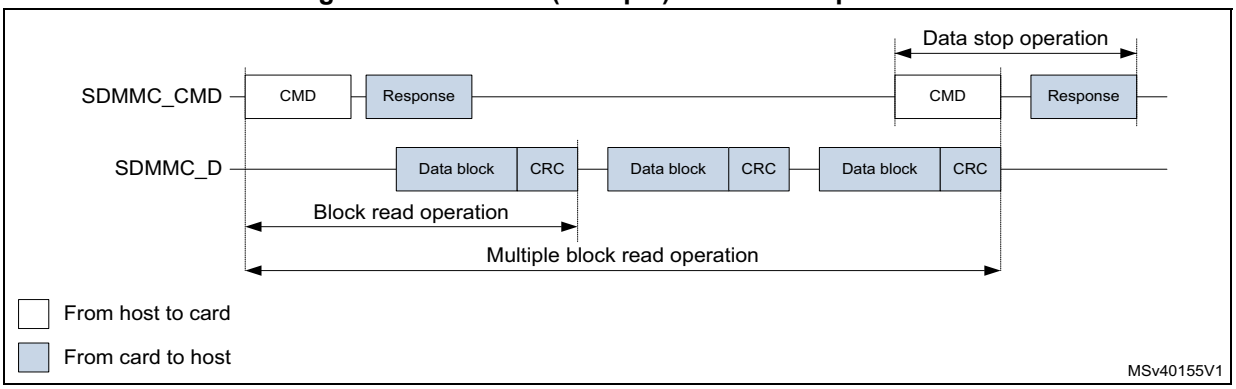
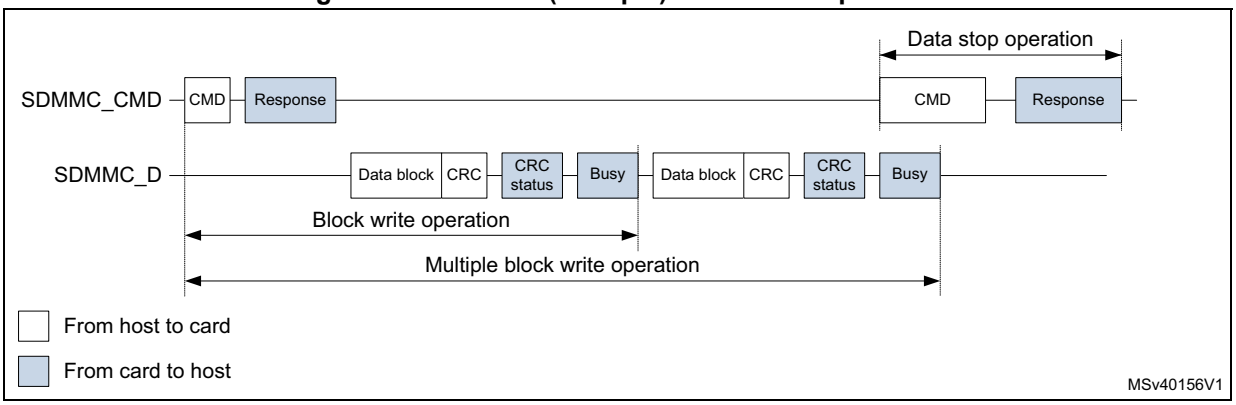


Figure 219. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 220. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 221. SDMMC (sequential) stream read operation

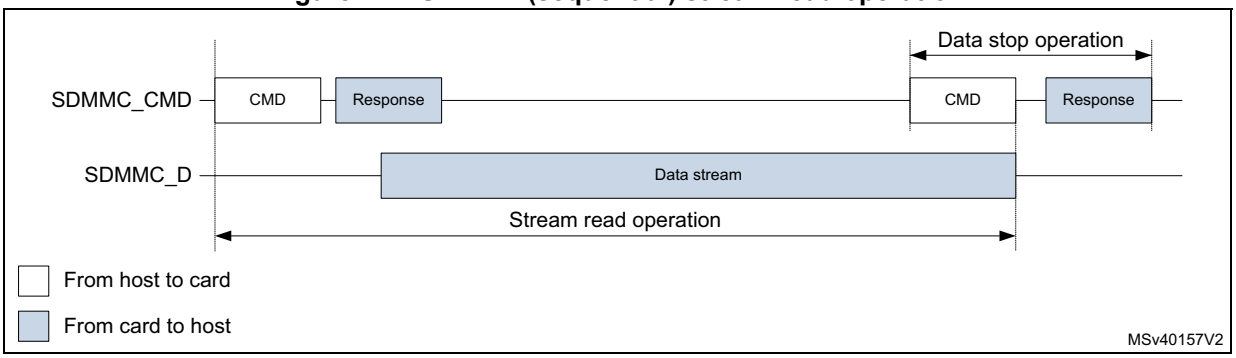
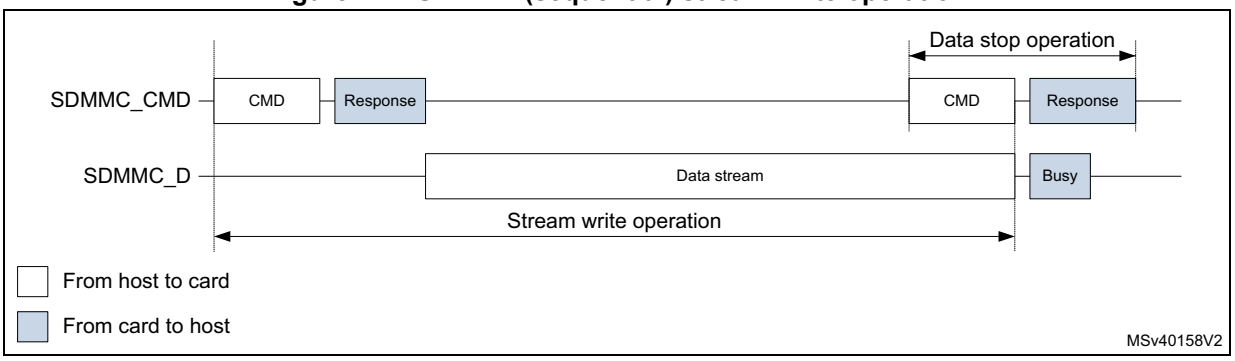


Figure 222. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

34.4 SDMMC operation modes

Table 271. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 272. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

34.5 SDMMC functional description

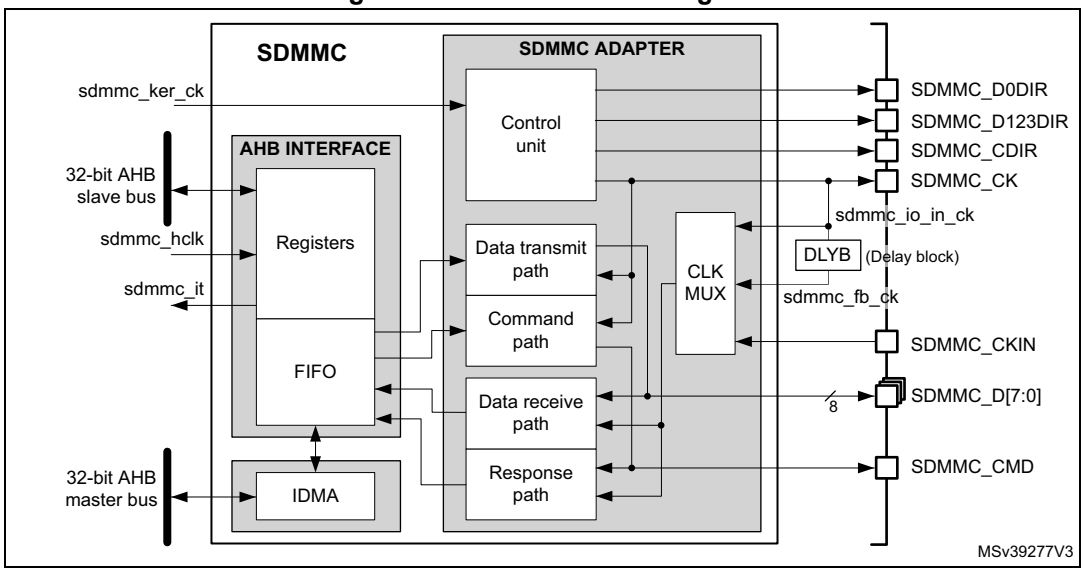
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

34.5.1 SDMMC block diagram

Figure 223 shows the SDMMC block diagram.

Figure 223. SDMMC block diagram



34.5.2 SDMMC pins and internal signals

Table 273 lists the SDMMC internal input/output signals, *Table 274* the SDMMC pins (alternate functions).

Table 273. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 273. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 274. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

34.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 224*.

Figure 224. SDMMC Command and data phase relation

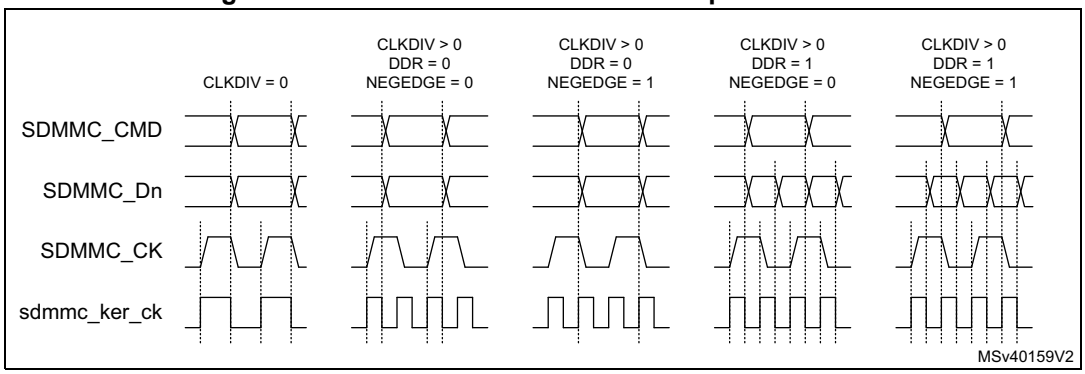


Table 275. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	Generated on sdmmc_ker_ck falling edge	
>0	0	0	Generated on sdmmc_ker_ck rising edge	Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

34.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 223: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

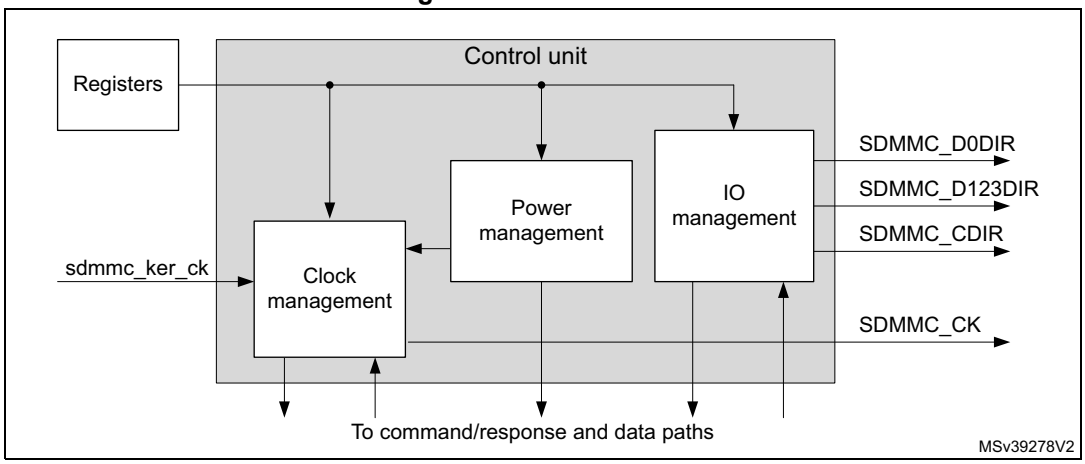
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 225*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 225. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

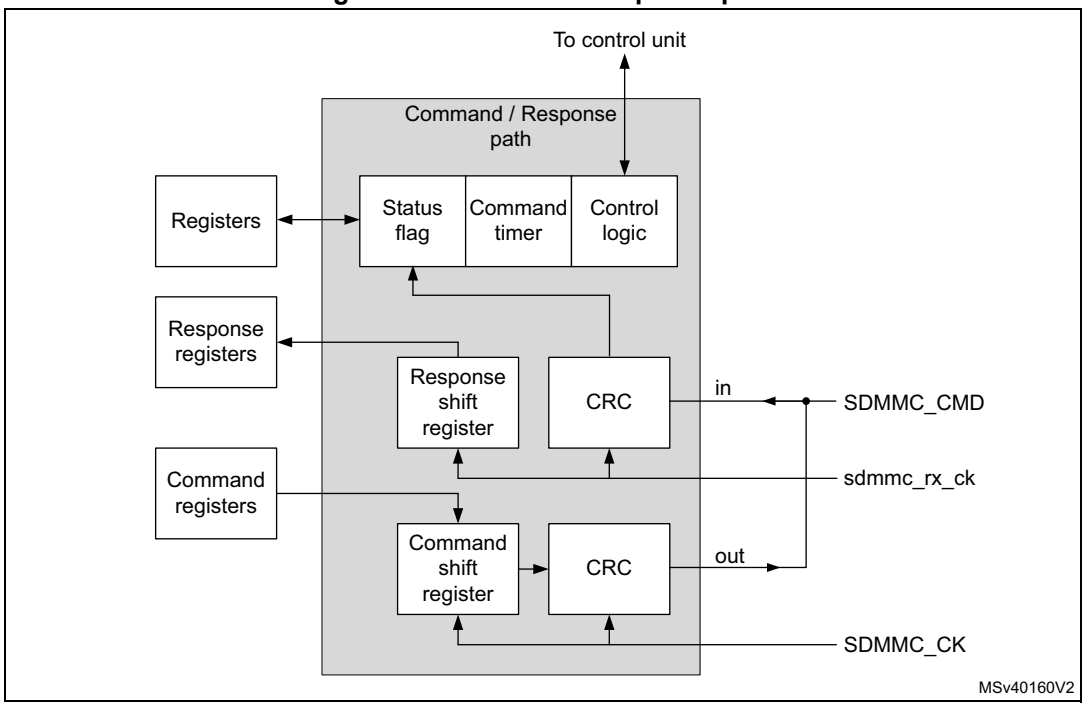
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 226. Command/response path

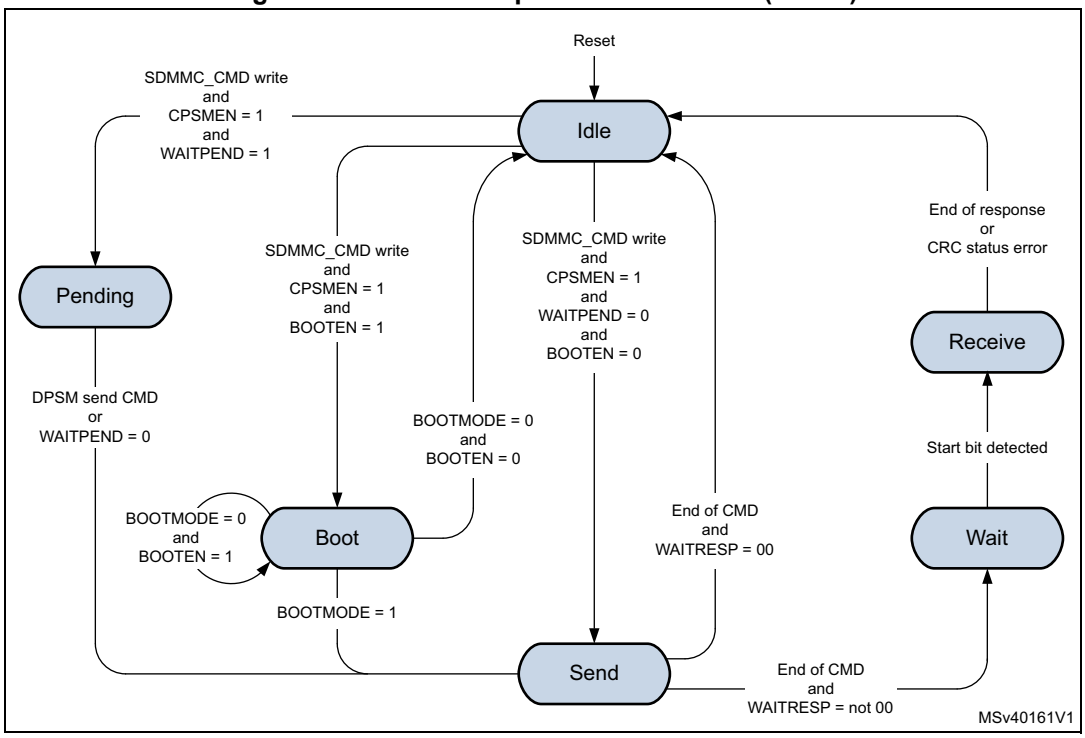


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 227. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 276.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 276. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 277*, *Table 278* and *Table 279*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 277. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 278. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 279. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 224*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder $[(M(x) * x^7) / G(x)]$

$G(x) = x^7 + x^3 + 1$

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit before CRC}) * x^0$

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 280*.

Table 280. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 281*:

Table 281. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 282*:

Table 282. Command path error handling

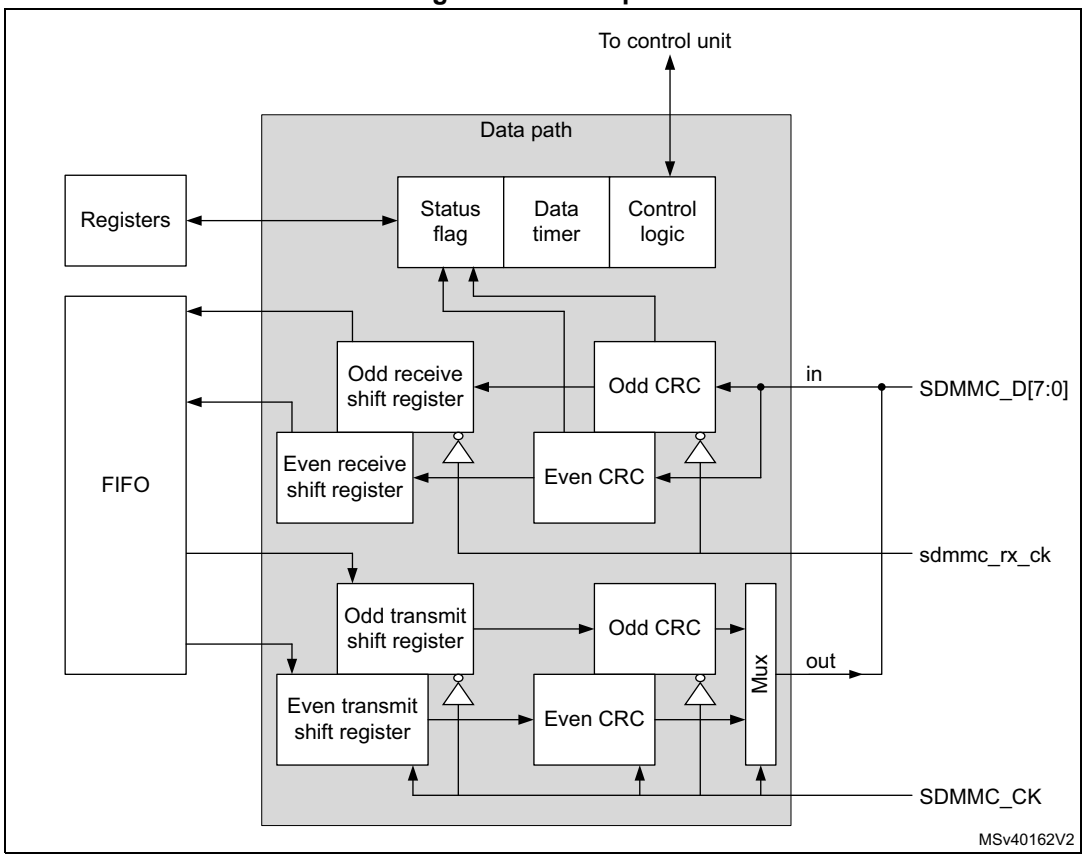
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 228* shows the data path block diagram.

Figure 228. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 229* and *Figure 230*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 229. DDR mode data packet clocking

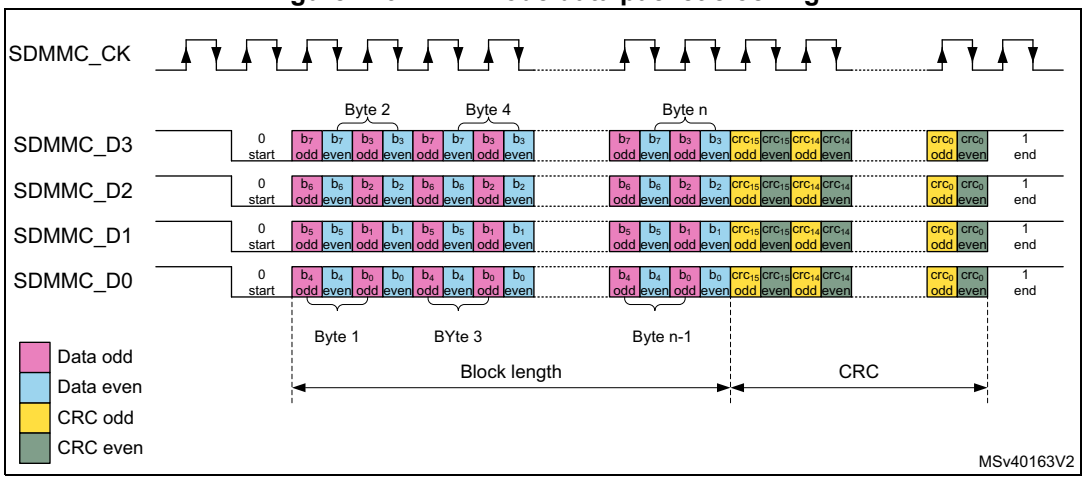
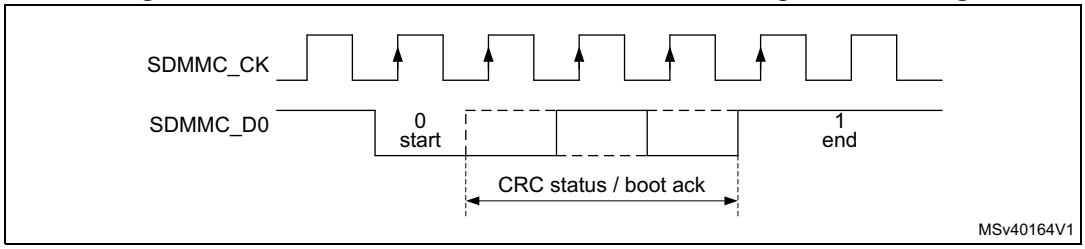


Figure 230. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

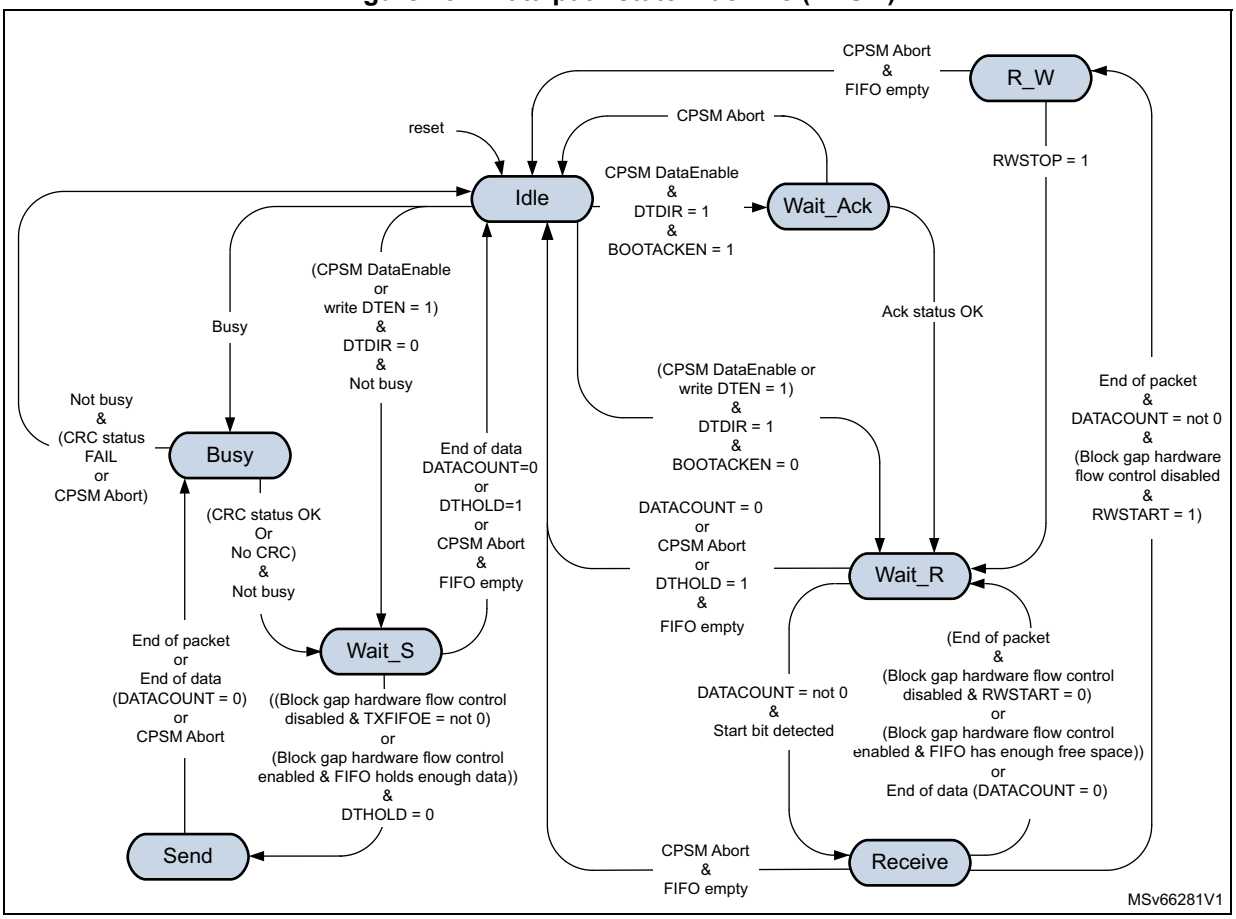
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 231*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 231. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.

- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.

– When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  – The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  – If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise, either immediately when block gap hardware flow control is disabled, or after enough free area is available in the FIFO when block gap hardware flow control is enabled.
  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  – In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  – When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  – When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.

  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0,
    - When block gap hardware flow control is disabled, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
    - When block gap hardware flow control is enabled, waits until the data FIFO hold enough data and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues

a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).

– If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.

– When receiving the CPSM Abort signal
- If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
- If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:

– If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
- it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
- When the card is busy SDMMC_D0 is low it remains in the Busy state.

– If a CRC status token is expected and indicates “erroneous transmission”.
- When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
- When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.

– If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.

– If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.

– When receiving the CPSM Abort signal in the Busy state:
- If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
- If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
- If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.

– When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.

– When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.

– When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 283. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 284*:

Table 284. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 284. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 285*:

Table 285. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 256-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 286*.

Table 286. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Enable hardware flow control HWFC_EN.
   If bus speed SDR50, SDR104, HS200 or DDR50 is used FIFO threshold can be used to provide flow control at the receive block gap.
3. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
4. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
5. When (DPSMACT = 1) write data to the FIFO.
   – When BUSSPEED = 0: The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
   - The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.
   – When BUSSPEED = 1 and the FIFO threshold is zero: The DPSM reads the transmit data from the FIFO.

- The SDMMC keeps transmitting data as long as FIFO is not empty, hardware flow control during the data transfer is used to prevent FIFO underrun.

– When BUSSPEED = 1 and the FIFO threshold is not zero: The DPSM reads the transmit data from the FIFO.
- The SDMMC only starts sending a new block when there is enough data in the FIFO according the threshold. Block gap hardware flow control is used to prevent FIFO underrun.

6. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
7. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 287*.

Table 287. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Enable hardware flow control HWFC_EN.
   – If bus speed SDR50, SDR104, HS200 or DDR50 is used FIFO threshold can be used to provide flow control at the receive block gap.
3. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
4. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
5. When (DPSMACT = 1) the FIFO is ready to receive data.
   – When BUSSPEED = 0: The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
   – When BUSSPEED = 1 and the FIFO threshold is zero: The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
   – When BUSSPEED = 1 and the FIFO threshold is not zero: The DPSM writes the received data to the FIFO.
   - The SDMMC only starts receiving a new block when there is enough free space in the FIFO according the threshold. Block gap hardware flow control is used to prevent FIFO overrun.
6. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
7. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 288*.

Table 288. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

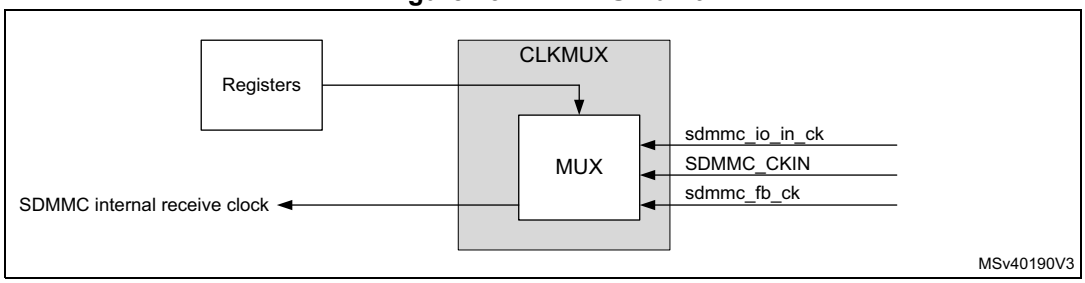
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 232. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

34.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer

multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

34.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 16 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer. When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

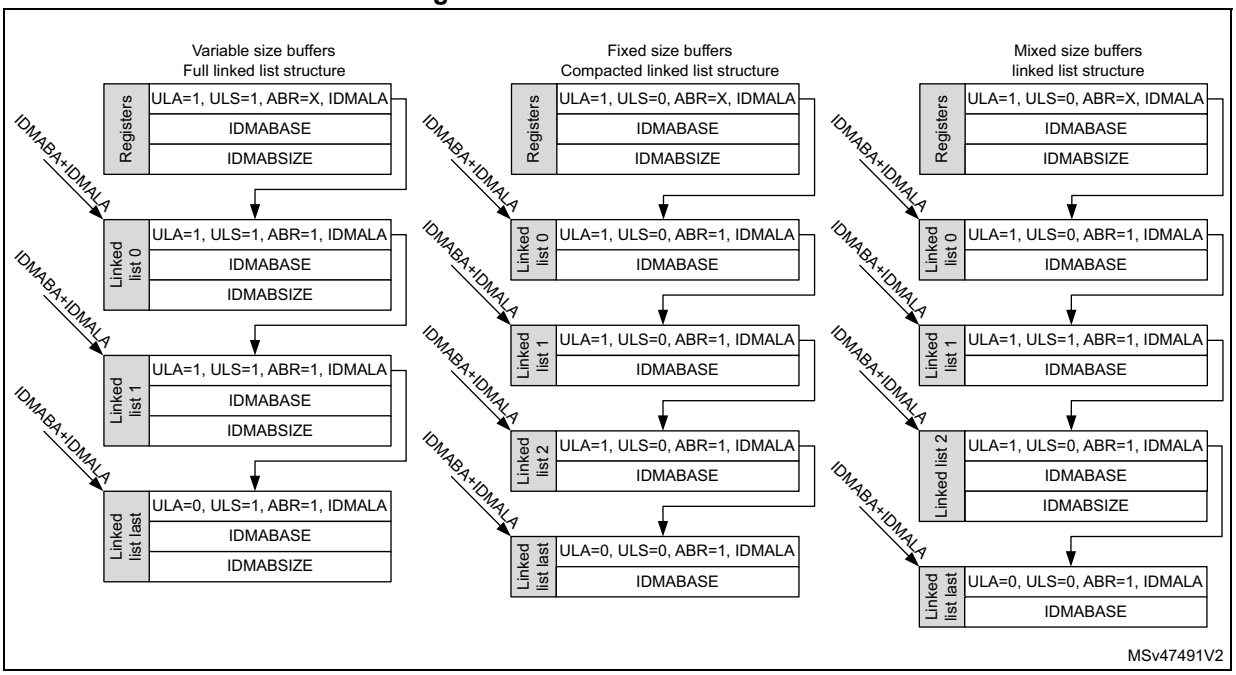
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 233. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

34.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have between 1.72x and 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50 Mbyte/s) the minimum sdmmc_hclk frequency is 21.43 MHz (85.72 Mbyte/s).

Table 289. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	13
e•MMC HS	8	52	26
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	100
SD DS / SDR12	4	25	5.36
SD HS / SDR25	4	50	10.72
SD DDR50	4	50	25

Table 289. AHB and SDMMC_CK clock frequency relation (continued)

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
SD SDR50	4	100	21.43
SD SDR104	4	208	44.58

Warning: When minimum HCLK frequency is not respected, overrun or underrun errors may occur even when hardware flow control is enabled.

34.6 Card functional description

34.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 290. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

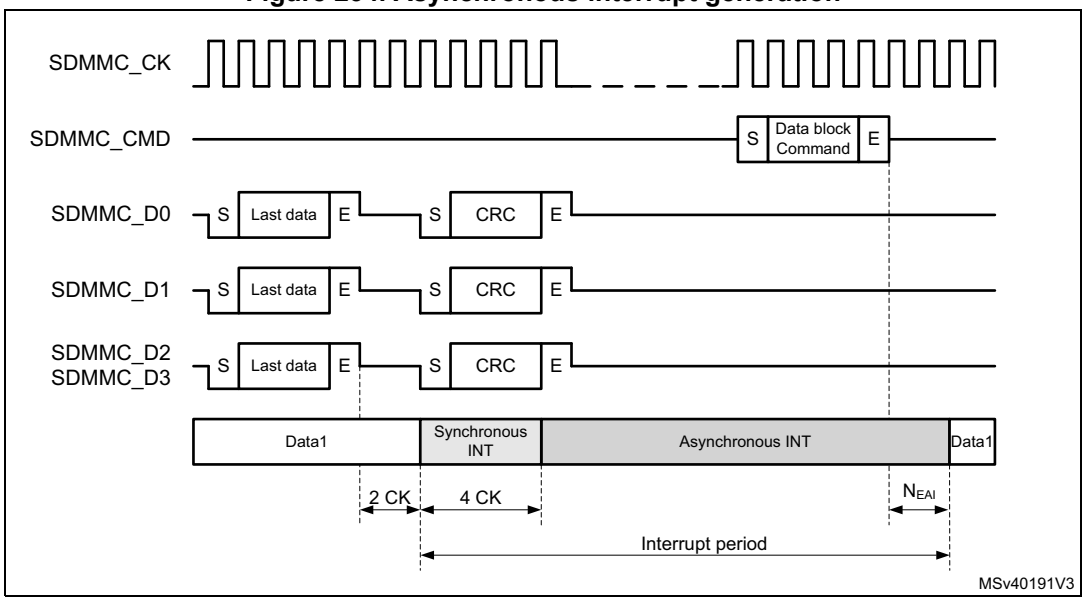
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 290*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 234. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 235. Synchronous interrupt period data read

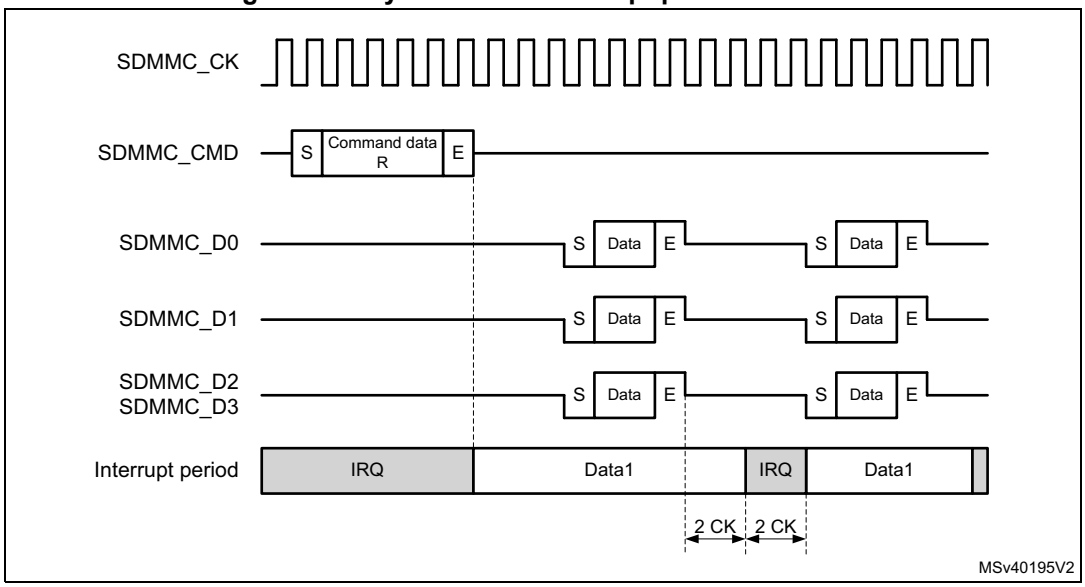
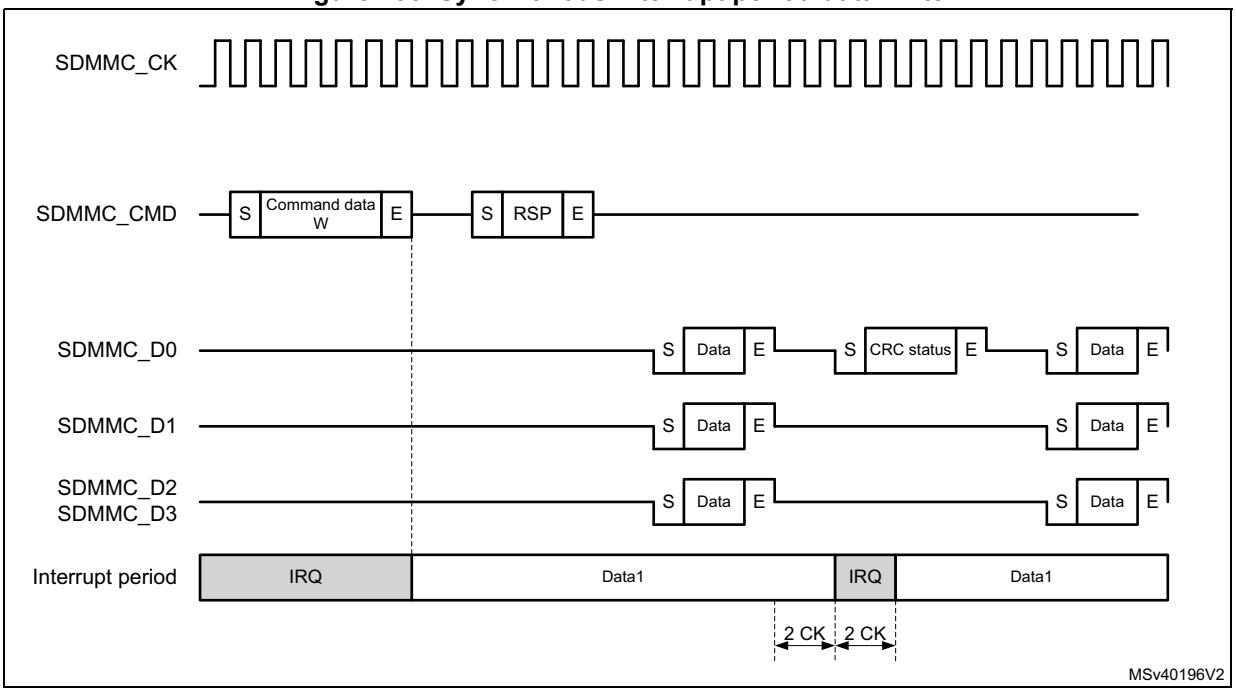


Figure 236. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a

command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.

- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 237. Asynchronous interrupt period data read

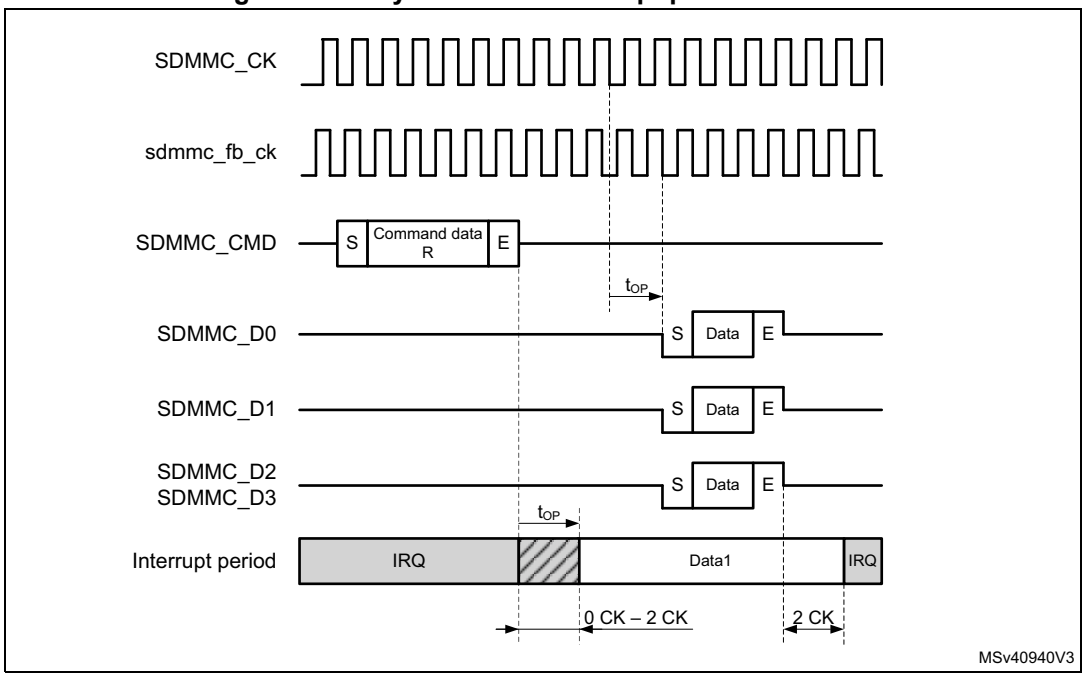
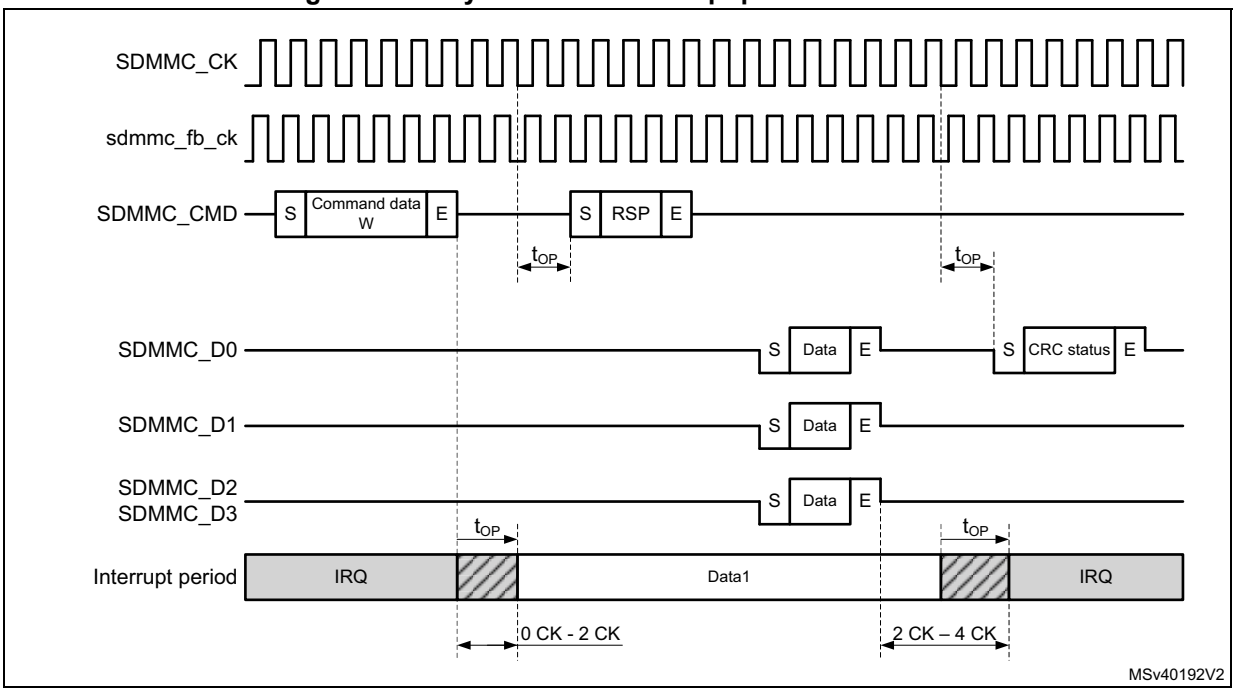


Figure 238. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 291*.

Table 291. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

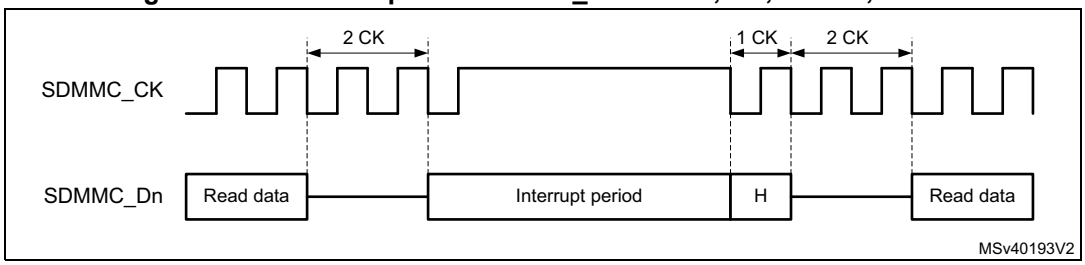
The SDMMC can perform a Read Wait with register settings according *Table 290*.

Read Wait is not available when block gap hardware flow control is enabled. (see *Section 34.7.2: Block gap hardware flow control*)

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

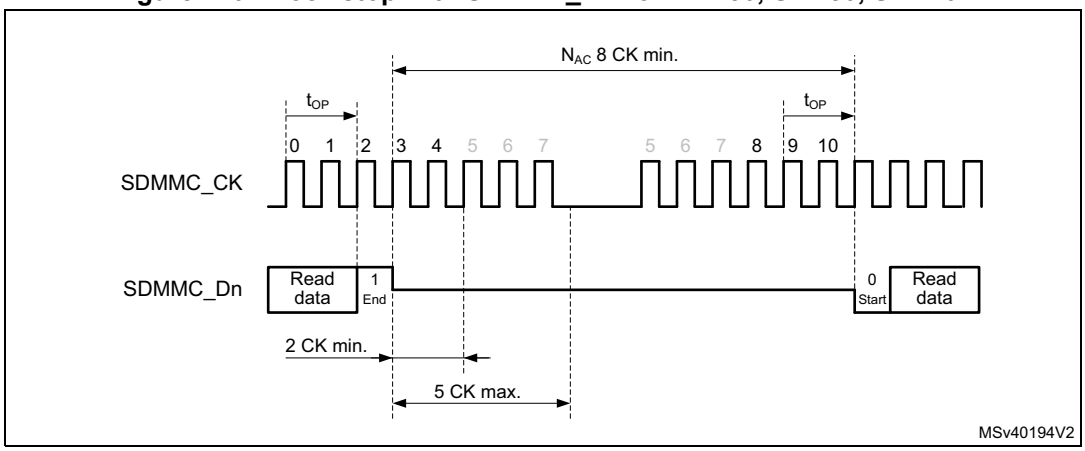
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 239*).

Figure 239. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 240*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 240. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



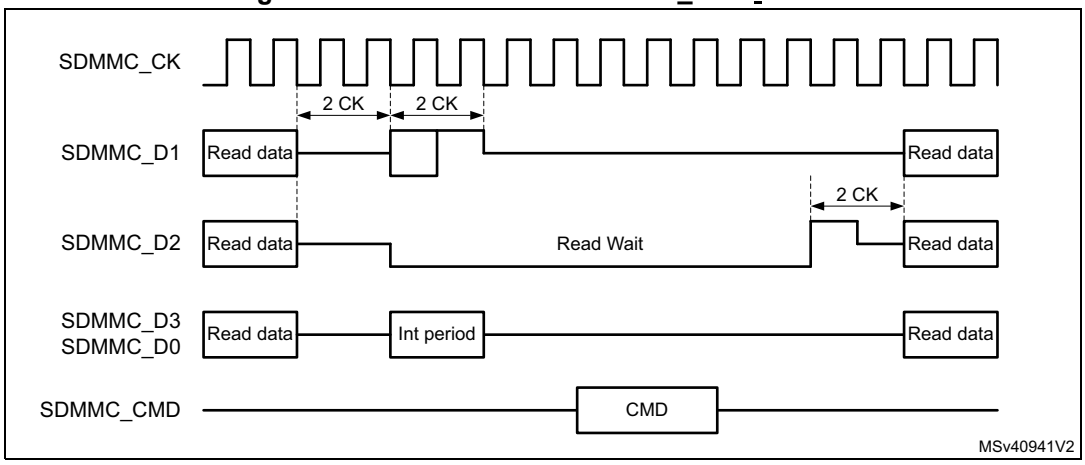
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

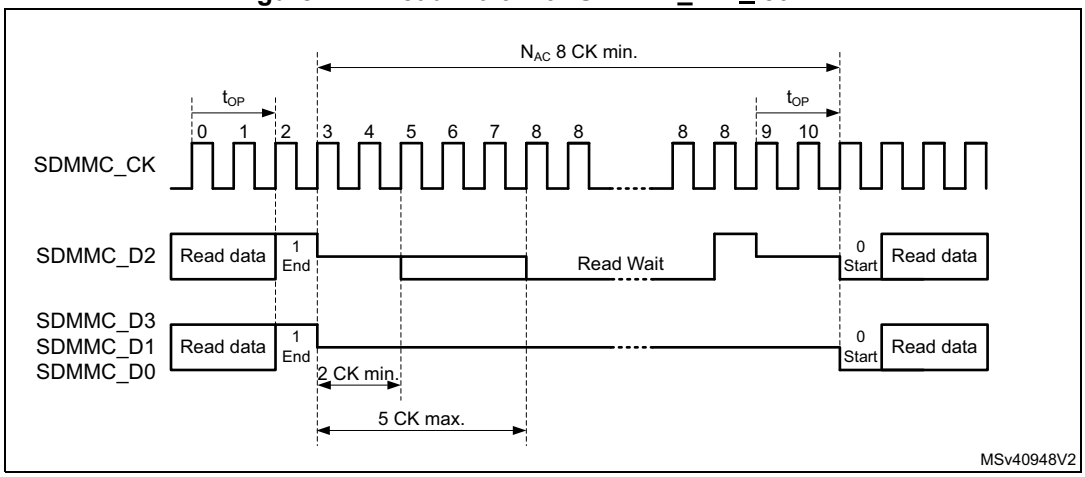
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 241*.

Figure 241. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 242*).

Figure 242. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

34.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 292. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

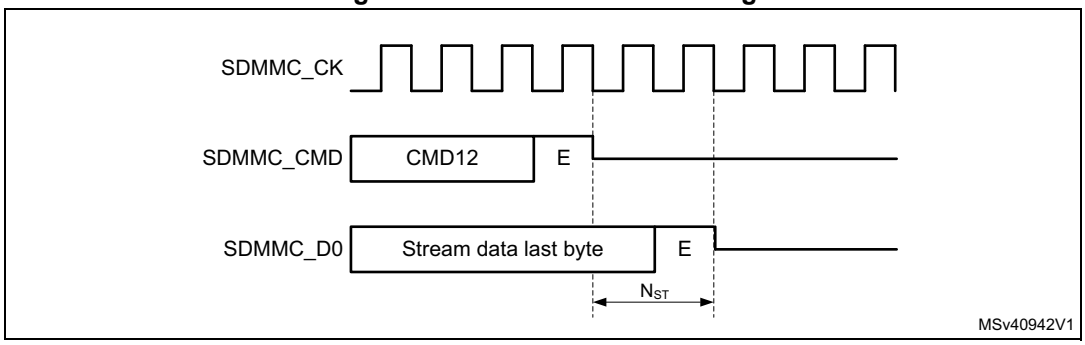
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 243. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

34.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

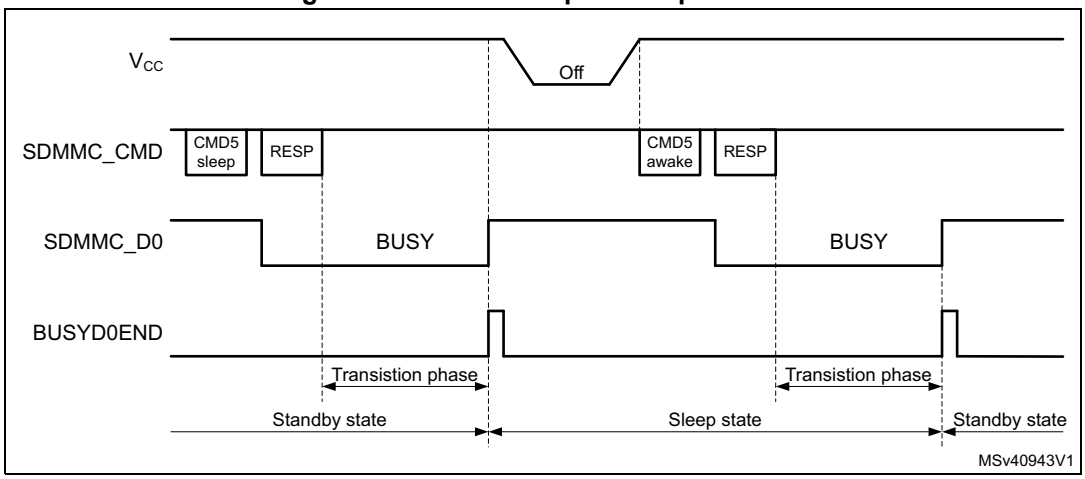
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 244. CMD5 Sleep Awake procedure



34.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

34.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

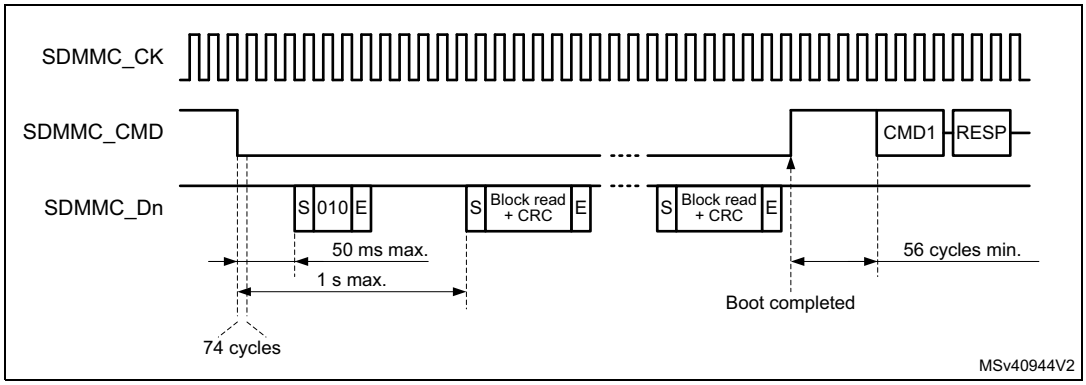
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 245. Normal boot mode operation



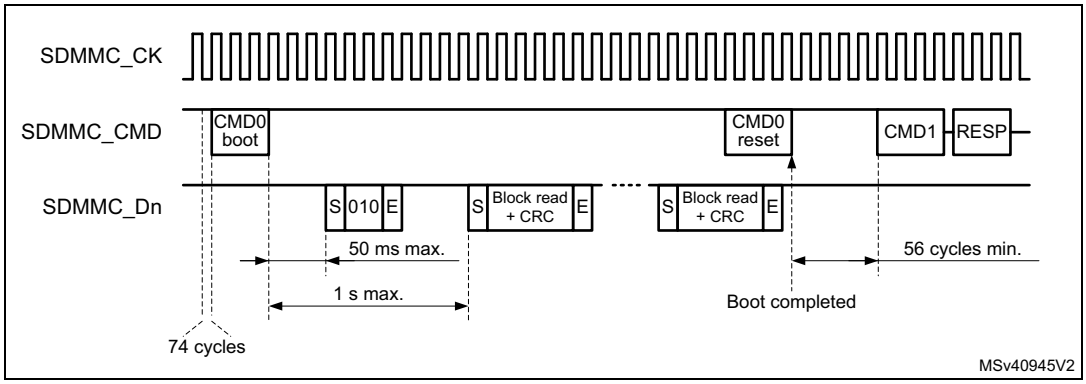
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   - If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 246. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

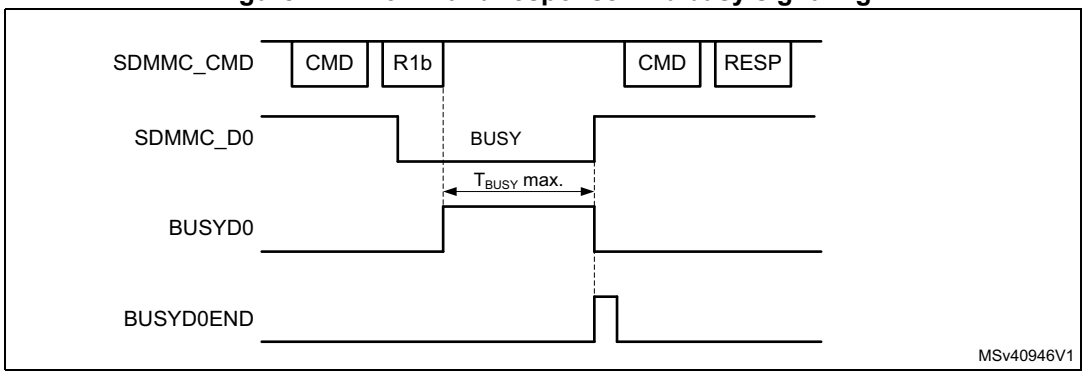
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

34.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 247. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

34.6.7 Reset and card cycle power

Reset

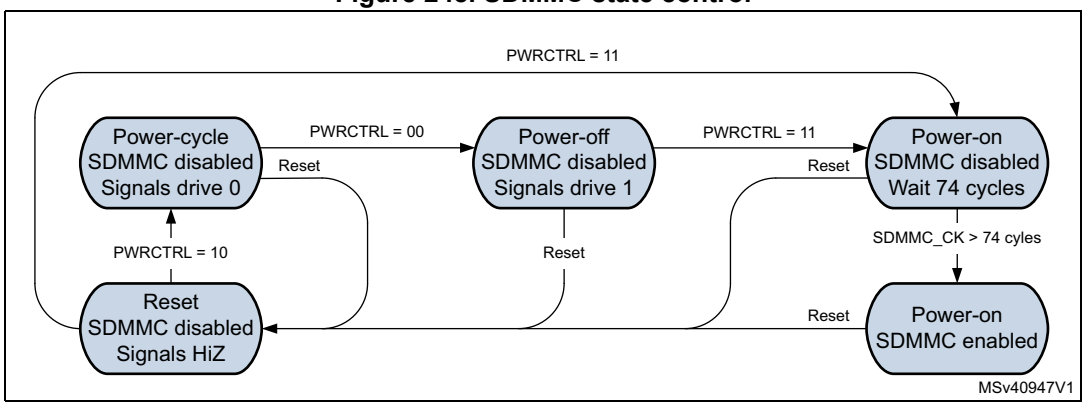
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 248.*.

Figure 248. SDMMC state control

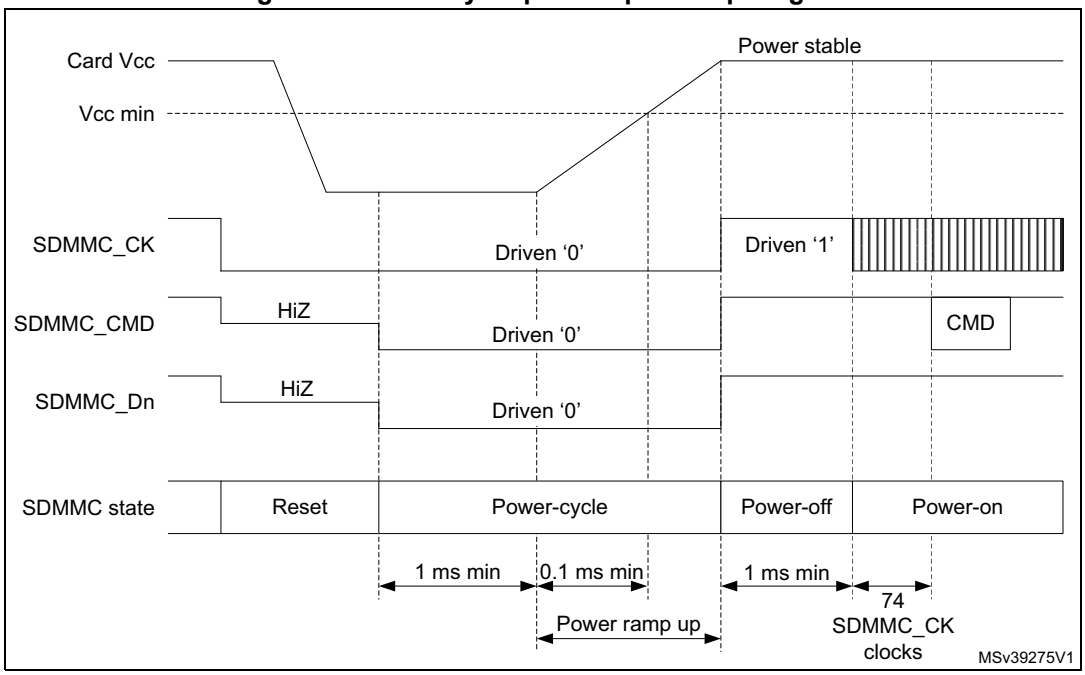


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 249. Card cycle power / power up diagram



34.7 Hardware flow control

There are two hardware flow control mechanisms:

1. Hardware flow control during data transfer:
   - Only to be used with cycle-aligned data transfers.
2. Block gap hardware flow control:
   - Only used in mode with bus speed SDR50, SDR104, HS200 and DDR50.
   - Must be used with variable delay.

34.7.1 Hardware flow control during data transfer

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control during data transfer must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control during data transfer can not be used and block gap hardware flow control must be used instead.

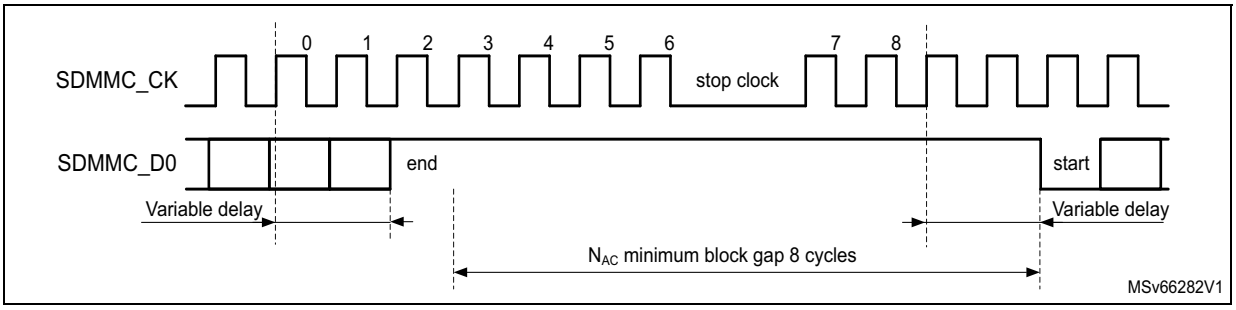
34.7.2 Block gap hardware flow control

The block gap hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors in SDR50, SDR104, HS200 and DDR50 mode (BUSSPEED = 1).

When transmitting, the behavior is to delay the start of a new write block and freeze the SDMMC state machines. The start of a new write block is delayed until the FIFO holds enough data according to the threshold. Only the state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled even if flow control is activated.

When receiving, the behavior is to stop SDMMC_CK during the read block gap and freeze the SDMMC state machines. The start of a new block transfer is stalled, due to stopped clock, until the FIFO has enough free space according to the threshold. Only the state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be emptied even if flow control is activated.

Figure 250. Read block gap hardware flow timing



Warning: When variable delay is used and block gap hardware flow control is enabled, sending commands during multiple block read may disturb the command response reception (CMD12 is not impacted by this).

Table 293. Data block gap hardware flow control behavior

Condition (HWFC_EN = 1 and BUSSPEED = 1)	Description
Data block size ≤ FIFO threshold	New block transfer is only started when the FIFO has enough data (TX mode) / free space (RX mode) up to or more than the THR threshold. *Note: Data flow control is guaranteed. A block transfer is only stared when enough data for at least one block is available / free in the FIFO*
Data block size > FIFO threshold	New block transfer is only started when the FIFO has enough data (TX mode) / free space (RX mode) up to or more than the THR threshold. *Note: Data flow control is not guaranteed. A block transfer is started even when not enough data for one block is available / free in the FIFO. The threshold can be used to trade off transfer latency / throughput versus FIFO underrun / overrun risk.*

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

The block gap hardware flow control is only enabled when HWFC_EN is set to 1, BUSSPEED = 1 and the THR FIFO threshold is different than 0. When BUSSPEED = 0 or FIFO threshold is set to 0, the hardware flow control is used during the data transfer. After reset, the hardware flow control is disabled.

Table 294. Hardware flow control selection

<table>
<tr><th>HWFC_EN</th><th>THR</th><th>BUSSPEED</th><th>Description</th></tr>
<tr><td>0</td><td>X</td><td rowspan="2">X</td><td>Hardware flow control disabled</td></tr>
<tr><td rowspan="3">1</td><td>0</td><td rowspan="2">Hardware flow control during data transfer enabled</td></tr>
<tr><td rowspan="2">not 0</td><td>0</td></tr>
<tr><td>1</td><td>Block gap hardware flow control enabled</td></tr>
</table>

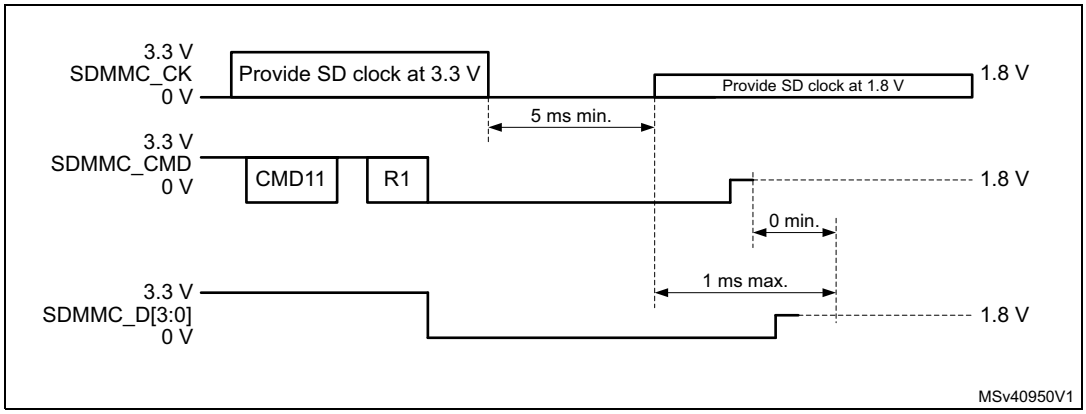
Block gap hardware flow control must be used when the SDMMC_Dn receive data are not cycle-aligned with the SDMMC_CK. This is whenever the sdmmc_fb_ck from the DLYB delay block is used, for example in the case of SDR104 receive mode with a t_{OP} and Dt_{OP} delay > 1 cycle.

34.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch

sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 251. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   – if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   – if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   – When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   – When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

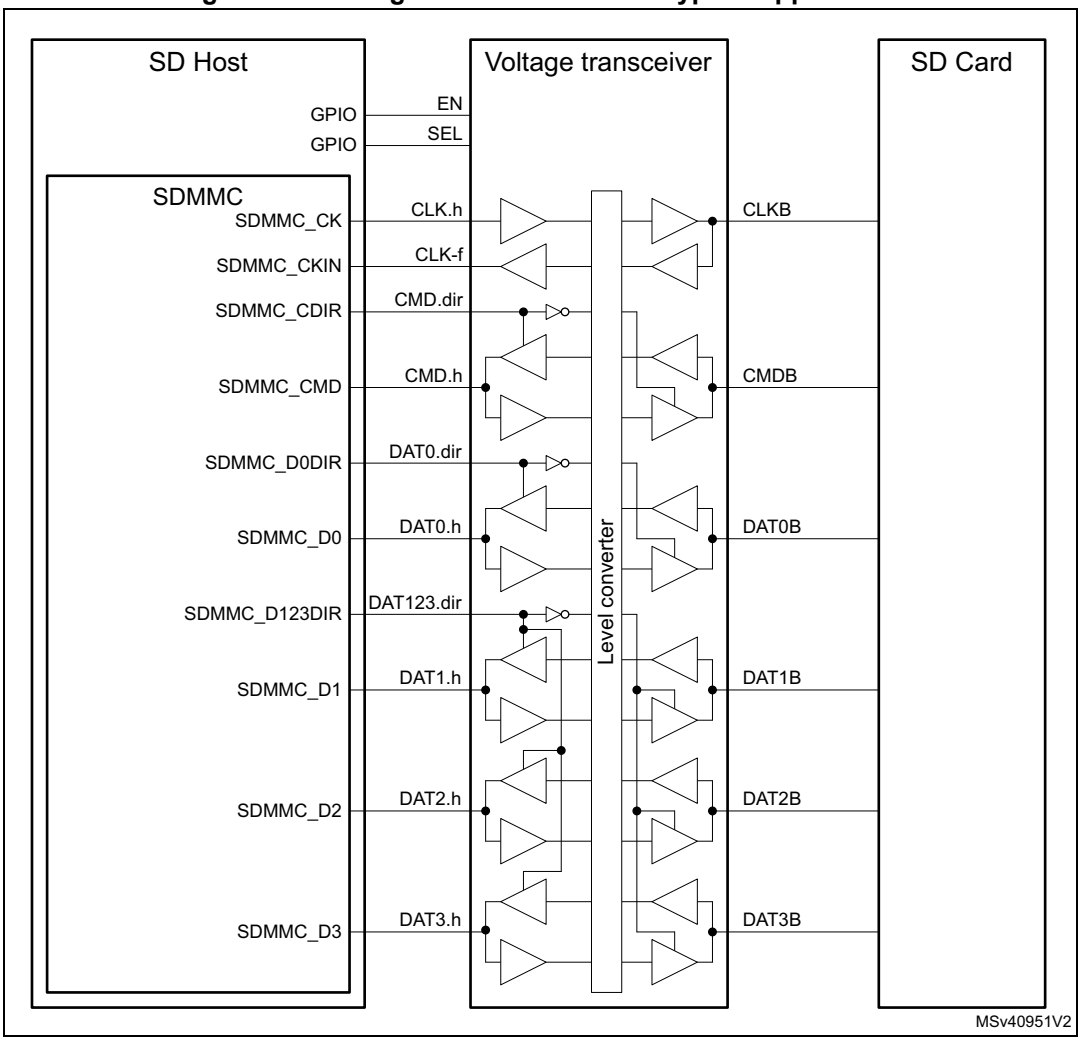
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 252. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

34.9 SDMMC interrupts

Table 295. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

34.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

34.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR POL	VSWI TCHEN	VSWI TCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

34.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 34.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

34.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

34.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

34.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

34.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTATUS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTATUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 296*.

The card status size is 32 or 128 bits, depending on the response type.

Table 296. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

34.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATATIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

34.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATALENGTH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

34.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: Block length = 2^0 = 1 byte
0001: Block length = 2^1 = 2 bytes
0010: Block length = 2^2 = 4 bytes
0011: Block length = 2^3 = 8 bytes
0100: Block length = 2^4 = 16 bytes
0101: Block length = 2^5 = 32 bytes
0110: Block length = 2^6 = 64 bytes
0111: Block length = 2^7 = 128 bytes
1000: Block length = 2^8 = 256 bytes
1001: Block length = 2^9 = 512 bytes
1010: Block length = 2^{10} = 1024 bytes
1011: Block length = 2^{11} = 2048 bytes
1100: Block length = 2^{12} = 4096 bytes
1101: Block length = 2^{13} = 8192 bytes
1110: Block length = 2^{14} = 16384 bytes
1111: Reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

34.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and

when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

34.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

34.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

34.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

34.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACKTIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

34.10.15 SDMMC data FIFO threshold register (SDMMC_FIFOTHRR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **THR[3:0]**: FIFO threshold
This bit can only be written by software when DPSM is inactive (DPSMACT = 0).
Define the data FIFO threshold:
0b0000:threshold = 0 byte (no threshold)
Others: threshold = $2^{(N-1)}$ bytes, threshold must not be programmed bigger than the FIFO size, 256 words.
Note: When THR > DBLOCKSIZE, blocks may not be transferred.

34.10.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IDMABMODE**: Buffer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

34.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[10]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABNDT[9:0]										Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:6 **IDMABNDT[10:0]**: Number of bytes per buffer

This 11-bit value must be multiplied by 16 to get the size of the buffer in 32-bit words and by 64 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 16 words = 64 bytes.
Example: IDMABNDT = 0x400: buffer size = 16384 words = 64 Kbytes.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 5:0 Reserved, must be kept at reset value.

34.10.18 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

34.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMALA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

34.10.20 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABA[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

34.10.21 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

34.10.22 SDMMC hardware configuration register 1 (SDMMC_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOSIZE[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **FIFOSIZE[3:0]**: SDMMC FIFO size

FIFO size = 2^{N+6} bytes, with N in the range 0 to 9

For FIFO size 2^6 bytes, the transfer beat is 8 (32 bytes)

For all other FIFO sizes, the transfer beat is 16 (64 bytes)

34.10.23 SDMMC version register (SDMMC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: SDMMC major revision number

Bits 3:0 **MINREV[3:0]**: SDMMC minor revision number

34.10.24 SDMMC identification register (SDMMC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0014 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IP_ID[31:0]**: SDMMC identification

34.10.25 SDMMC size identification register (SDMMC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: SDMMC size identification

34.10.26 SDMMC register map

Table 297. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_ RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_ RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_ DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_ DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]			DTMODE[1:0]		DTDIR	DTEN	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 297. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	SDMMC_ DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_ STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SDMMC_ FIFOTHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR[3:0]			
	Reset value																													0	0	0	0
0x048 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[10:0]											Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0						
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 297. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0 - 0x3EC	Reserved	Res.																															
0x3F0	SDMMC_ HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOSIZE [3:0]			
	Reset value																													0	1	0	0
0x3F4	SDMMC_ VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	1	0	0	0	0
0x3F8	SDMMC_ IPIDR	IP_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0x3FC	SDMMC_ SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

35 Delay block (DLYB)

35.1 DLYB introduction

The delay block (DLYB) is used to generate two output clocks that are dephased from 2 input clocks. The phase of each output clock must be programmed by the user application. The output clocks are then used to clock the data transmitted/received by another peripheral such as an SDMMC or Octo-SPI interface.

35.2 DLYB main features

The delay block has the following features:

- Frequency range for Lock mode is [50 MHz, 208 MHz]
- 32 phases (delay taps)
- Bypass mode for operation below 50 MHz

35.3 DLYB implementation

Table 298. STM32MP21xx features

DLYB associated peripheral	DLYBOS1	DLYBSD1	DLYBSD2	DLYBSD3
OCTOSPI1	X	-	-	-
SDMMC1	-	X	-	-
SDMMC2	-	-	X	-
SDMMC3	-	-	-	X

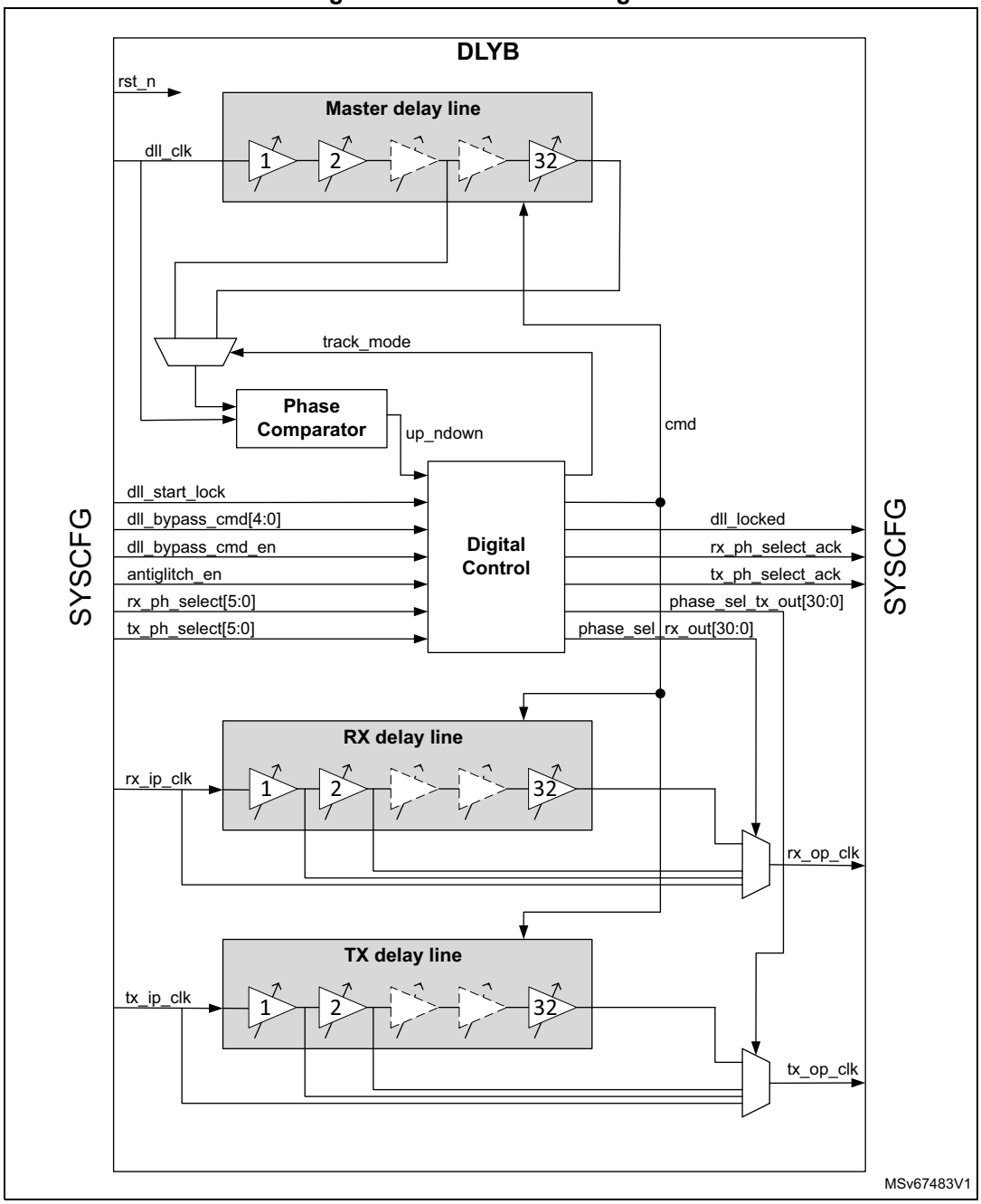
35.4 DLYB functional description

35.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- Digital block to generate commands for delay lines and phase selection. This block embeds lock FSM.
- Delay line supporting the unit delays
- Output clock selection multiplexer

Figure 253. DLYB block diagram



35.4.2 DLYB internal signals

Table 299 lists the DLYB functional signals.

Table 299. DLYB functional signals

Signal name	Signal type	Description
dll_clk	Digital input	Free-running input clock used as reference clock for the Master delay line and FSM blocks
tx_ip_clk	Digital input	Input clock of the TX slave delay line
rx_ip_clk	Digital input	Input clock of the RX slave delay line
tx_op_clk	Digital output	Selected output clock of the TX slave delay line
rx_op_clk	Digital output	Selected output clock of the RX slave delay line
rst_n	Digital input	Asynchronous active low reset
dll_start_lock	Digital input	Enable of DLL when configured in Lock mode
dll_locked	Digital output	Lock indicator. When high, it indicates that the FSM has finished the lock sequence. The DLL can be used
tx_ph_select[5:0]	Digital input	Selection of the slave DLL phase shift over 33 for TX path 000000: 0 tap selection 000001: 1 tap selection 011111: 31 taps selection 1xxxxx: 32 taps selection
tx_ph_select_ack	Digital output	When high: phase selected by tx_ph_select is acknowledged When low: output clock tx_op_clk is tied low
rx_ph_select[5:0]	Digital input	Selection of the slave DLL phase shift over 33 for RX path 000000: 0 tap selection 000001: 1 tap selection 011111: 31 taps selection 1xxxxx: 32 taps selection
rx_ph_select_ack	Digital output	When high: phase selected by rx_ph_select is acknowledged When low: output clock rx_op_clk is tied low
antiglitch_en	Digital input	When high: output clocks (tx_op_clk or rx_op_clk) when minimum delay is selected (tx_ph_select = 0 or rx_ph_select=0) are gated to avoid any glitch when phase selection changes When low: output clocks (tx_op_clk or rx_op_clk) when minimum delay is selected (tx_ph_select = 0 or rx_ph_select=0) are not gated (possible glitches if input clocks (tx_ip_clk or rx_ip_clk) are enabled when phase selection changes)
dll_bypass_cmd_en	Digital input	Active high: use dll_bypass_cmd for command generation
dll_bypass_cmd[4:0]	Digital input	permits to force delays for all taps when dll_bypass_cmd_en high: 00000: not functional 00001: pass gate 0 is activated (min delay) 00010: pass gate 1 is activated 11000: pass gate 23 is activated 11001 to 11111: pass gate 23 is activated (max delay)

Table 300 lists the DLYB interconnections. The SYSCFG registers are highlighted in light gray, RCC registers in dark gray and peripheral ports in blue.

Table 300. SDMMC DLYB interconnections

Signal name	Source/destination		
	SDMMC1	SDMMC2	SDMMC3
dll_clk	ClkDll (sdmmc1_ClkDll)	ClkDll (sdmmc2_ClkDll)	ClkDll (sdmmc3_ClkDll)
rx_ip_clk	CKio (SDMMC1_CK_in)	CKio (SDMMC2_CK_in)	CKio (SDMMC3_CK_in)
tx_ip_clk	Not used		
rx_op_clk	FB_CLK (sdmmc1_fb_clk)	FB_CLK (sdmmc2_fb_clk)	FB_CLK (sdmmc3_fb_clk)
tx_op_clk	Not used		
rst_n	RCC_SDMMC1CFGR. SDMMC1DLLRST	RCC_SDMMC2CFGR. SDMMC2DLLRST	RCC_SDMMC3CFGR. SDMMC3DLLRST
dll_start_lock	SYSCFG_DLYBSD1CR.EN	SYSCFG_DLYBSD2CR.EN	SYSCFG_DLYBSD3CR.EN
dll_locked	SYSCFG_DLYBSD1SR. LOCK	SYSCFG_DLYBSD2SR. LOCK	SYSCFG_DLYBSD3SR. LOCK
tx_ph_select[5:0]	Not used		
tx_ph_select_ack	Not used		
rx_ph_select[5:0]	SYSCFG_DLYBSD1CR. RX_TAP_SEL[5:0]	SYSCFG_DLYBSD2CR. RX_TAP_SEL[5:0]	SYSCFG_DLYBSD3CR. RX_TAP_SEL[5:0]
rx_ph_select_ack	SYSCFG_DLYBSD1SR. RX_TAP_SEL_ACK	SYSCFG_DLYBSD2SR. RX_TAP_SEL_ACK	SYSCFG_DLYBSD3SR. RX_TAP_SEL_ACK
antiglitch_en	SYSCFG_DLYBSD1CR. ANTIGLITCH_EN	SYSCFG_DLYBSD2CR. ANTIGLITCH_EN	SYSCFG_DLYBSD3CR. ANTIGLITCH_EN
dll_bypass_cmd_en	SYSCFG_DLYBSD1CR. BYP_EN	SYSCFG_DLYBSD2CR. BYP_EN	SYSCFG_DLYBSD3CR. BYP_EN
dll_bypass_cmd[4:0]	SYSCFG_DLYBSD1CR. BYP_CMD[4:0]	SYSCFG_DLYBSD2CR. BYP_CMD[4:0]	SYSCFG_DLYBSD3CR. BYP_CMD[4:0]

Table 301. OCTOSPI DLYB interconnections

Signal name	Source/destination
	OCTOSPI1
dll_clk	ospi_clk_o_free_run
rx_ip_clk	ospi_clk_fb_to_dly_o
tx_ip_clk	ospi_clk_o
rx_op_clk	ospi_clk_fb_dly_i
tx_op_clk	ospi_clk_ddrohold_dly_i

Table 301. OCTOSPI DLYB interconnections (continued)

Signal name	Source/destination
	OCTOSPI1
rst_n	RCC_OSPI1CFGR.OSPI1DLLRST
dll_start_lock	SYSCFG_DLYBOS1CR.EN
dll_locked	SYSCFG_DLYBOS1SR.LOCK
tx_ph_select[5:0]	SYSCFG_DLYBOS1CR.TX_TAP_SEL[5:0]
tx_ph_select_ack	SYSCFG_DLYBOS1SR.TX_TAP_SEL_ACK
rx_ph_select[5:0]	SYSCFG_DLYBOS1CR.RX_TAP_SEL[5:0]
rx_ph_select_ack	SYSCFG_DLYBOS1SR.RX_TAP_SEL_ACK
antiglitch_en	Not used (disabled)
dll_bypass_cmd_en	SYSCFG_DLYBOS1CR.BYP_EN
dll_bypass_cmd[4:0]	SYSCFG_DLYBOS1CR.BYP_CMD[4:0]

35.4.3 General description

The delay block is structured around one master delay line and two slave delay lines (RX and TX) and there are two operating modes:

- Lock mode (*dll_bypass_cmd_en* = 0)
  - This mode can only be used when the frequency of dll_clk is in the range [50 MHz, 208 MHz]
  - The delay block is enabled by setting *dll_start_lock* to ‘1’.
  - The delay block locks on the frequency of the *dll_clk* input clock and compensates for voltage and temperature variations. This process determines the command to be applied to each of the 32 delay units (a.k.a. tap) of the master delay line for the total master delay to represent one period of *dll_clk*. This command is copied to each delay unit of the RX and TX delay lines. The command varies with voltage and temperature for the DLL to remain locked. This guarantees that each delay unit of the RX and TX delay lines represents 1/32th of the input clock period.
  - Independent delay values can be selected for the RX and TX delay lines by controlling *rx_ph_select[5:0]* and *tx_ph_select[5:0]* parameters. The granularity of the delay is 1/32th of the input clock period.
- Bypass mode (*dll_bypass_cmd_en* = 1)
  - This mode is required to be able to insert a delay between the *rx_ip_clk* (resp. *tx_ip_clk*) input clock and the *rx_op_clk* (resp. *tx_op_clk*) output clock when the operating frequency is lower than the minimum frequency supported by the delay block in lock mode (frequency < 50 MHz). This is necessary to support OCTOSPI devices in DTR mode at low frequency for which DQS must be delayed to sample data.
  - In this mode, delays in the RX and TX delay lines are used as “pure” delays, i.e. without trying to lock on the frequency of the *dll_clk* input clock. There is therefore no voltage nor temperature compensation in this mode and the RX and TX delay values vary with voltage and temperature. The command applied to each delay unit of the RX and TX delay lines is controlled by *dll_bypass_cmd[4:0]*. Each delay

unit includes 24 delay elements. The number of delay elements which are enabled in each delay unit depends on the value of *dll_bypass_cmd[4:0]*.

– Independent delay values can be selected for the RX and TX delay lines by controlling *rx_ph_select[5:0]* and *tx_ph_select[5:0]* parameters. The total delay inserted by the RX (resp. TX) delay line depends on *rx_ph_select[5:0]* (resp. *tx_ph_select[5:0]*) and *dll_bypass_cmd[4:0]* values.

35.4.4 Lock mode procedure

For frequencies greater or equal to 50MHz, procedure is given below:

1. When *rst_n* is low, DLYB lock FSM is put in Idle mode. Note that *dll_start_lock* must be set to ‘0’ during hard reset.
2. Once the input signal *dll_clk* is stable (after a frequency switch for example) the lock sequence can be started by setting *dll_start_lock* to ‘1’ and maintain this value.
3. During lock sequence DLYB can not be used until *dll_locked* flag is set to ‘1’.
4. Once it is done, delay line for master and slaves are PVT compensated (tracking V and T evolution but not a frequency switch).
5. Then the phase can be programmed using *tx_ph_select[5:0]* and *rx_ph_select[5:0]* input signals. The selected delays are applied to TX output clock and RX output clock once *tx_ph_select_ack* and *rx_ph_select_ack* are set to ‘1’ respectively.
6. If another lock sequence is needed, for a frequency switch for example, *dll_start_lock* signal must go to ‘0’ and sequence can be restarted with step 2.

35.4.5 Bypass mode procedure

For frequencies lower the 50 MHz, procedure is given below:

1. When *rst_n* is low, DLYB lock FSM is put in Idle mode. Note that *dll_start_lock* must be set to ‘0’ during hard reset.
2. Delay block is enabled by setting *dll_bypass_cmd_en* to ‘1’. The command applied to every delay unit is defined by *dll_bypass_cmd[4:0]* input (the value 0 is reserved and must not be used).
3. The phase can be programmed using *tx_ph_select[5:0]* and *rx_ph_select[5:0]* input signals. The selected delays are applied to TX output clock and RX output clock once *tx_ph_select_ack* and *rx_ph_select_ack* are set to ‘1’ respectively. For minimum delay, *tx_ph_select[5:0]* (resp. *rx_ph_select[5:0]*) must be set to 0.

36 Analog-to-digital converters (ADC)

36.1 ADC introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are controlled independently.

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

36.2 ADC main features

- High-performance features
  - Up to 2 ADCs
    ADC1 is connected to 18 external channels + 2 internal channels
    ADC2 is connected to 14 external channels + 5 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Management of single-ended and differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Offset calibration support
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Data can be routed to MDF for post processing
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency

  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 18 GPIO pads
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel connected to the analog positive reference voltage V_{REF+}
  - 1 channel for monitoring V_{DDCORE} internal voltage
  - 1 channel for monitoring V_{DDCPU} internal voltage
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 254 shows the block diagram of one ADC.

36.3 ADC implementation

Table 302. ADC features

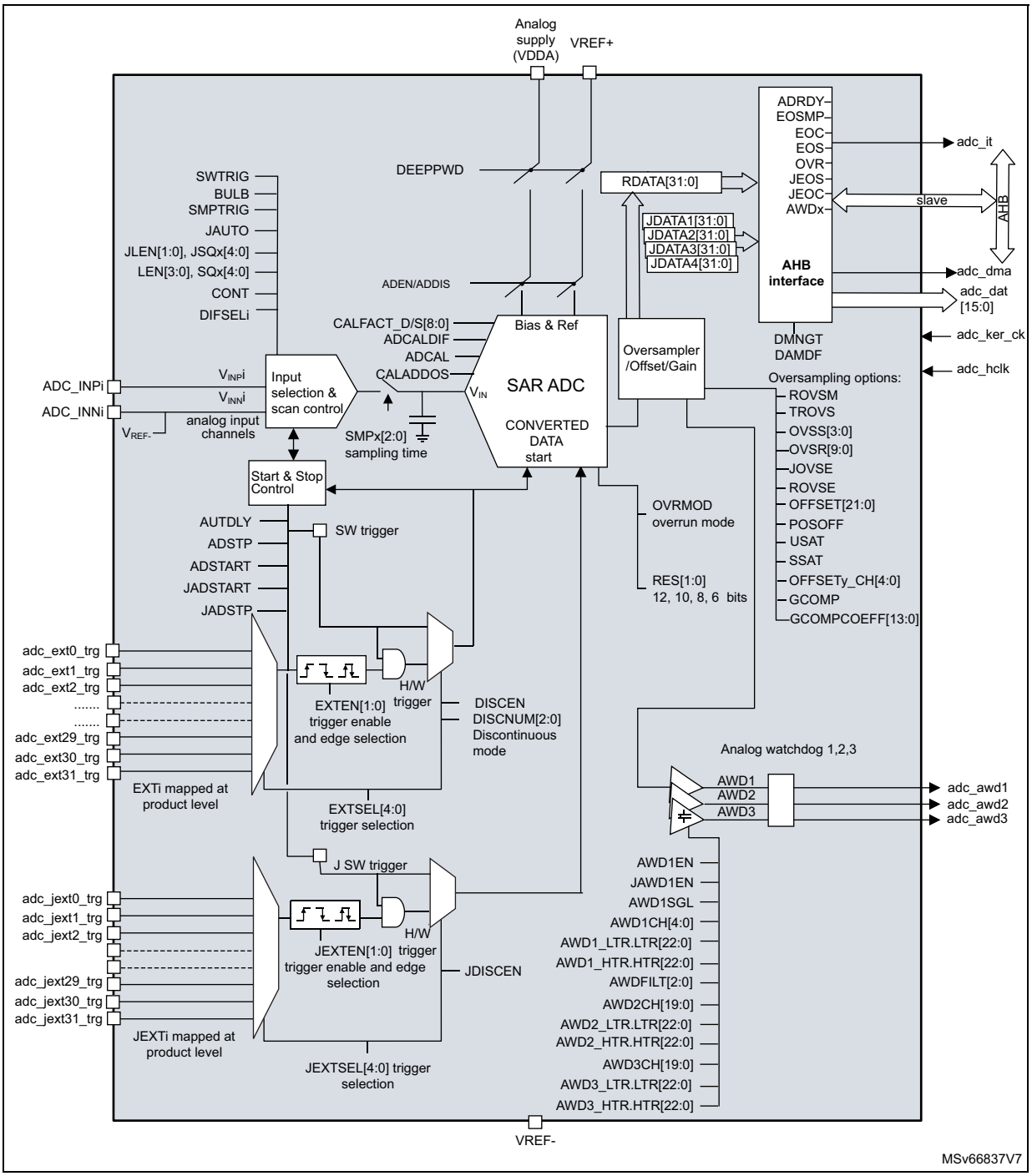
ADC modes/features	ADC1	ADC2
Resolution	12 bits	
Maximum sampling speed	5 Msps (12-bit resolution)	
Dual mode operation	-	
Offset calibration	X	
Single-end input	X	
Differential input	X	
Injected channel conversion	X	
Oversampling	up to x1024	
Data register	32 bits	
DMA support	X	
Parallel data output to MDF	X	
Offset compensation	X	
Gain compensation	X	
Number of analog watchdog	3	

36.4 ADC functional description

36.4.1 ADC block diagram

Figure 254 shows the ADC block diagram and *Table 303* gives the ADC pin description.

Figure 254. ADC block diagram



36.4.2 ADC pins and internal signals

Table 303. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
ADCx_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 36.4.4: ADC connectivity* for details)

Table 304. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADCx_INN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3)
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
adc_dat[15:0]	Output	ADC data outputs

Table 305. ADC1 interconnection

Signal name	Source/destination
ADC1 V_{INP}[14]	V_{REFINT} (output voltage from internal reference voltage)
ADC1 V_{INP}[19]	V_{REF+}
adc_dat1[15:0]	mdf_adc1_dat[15:0]
adc_ext_trg0	tim1_trgo
adc_ext_trg1	tim1_trgo2
adc_ext_trg2	tim8_trgo
adc_ext_trg3	tim8_trgo2

Table 305. ADC1 interconnection

Signal name	Source/destination
adc_ext_trg4	exti12 (from EXTI1)
adc_ext_trg5	Reserved
adc_ext_trg6	tim2_trgo
adc_ext_trg7	tim3_trgo
adc_ext_trg8	tim4_trgo
adc_ext_trg9	tim5_trgo
adc_ext_trg10	tim6_trgo
adc_ext_trg11	tim15_trgo
adc_ext_trg12	tim1_oc1
adc_ext_trg13	tim1_oc2
adc_ext_trg14	tim1_oc3
adc_ext_trg15	Reserved
adc_ext_trg16	Reserved
adc_ext_trg17	Reserved
adc_ext_trg18	tim2_oc2
adc_ext_trg19	tim3_oc4
adc_ext_trg20	tim4_oc4
adc_ext_trg21	tim5_oc1
adc_ext_trg22	tim12_oc1
adc_ext_trg23	exti11 (from EXTI2)
adc_ext_trg24	lptim1_ch1
adc_ext_trg25	lptim2_ch1
adc_ext_trg26	lptim3_ch1
adc_ext_trg27	lptim4_ch1
adc_ext_trg28	lptim5_out
adc_ext_trg29	Reserved
adc_ext_trg30	Reserved
adc_ext_trg31	Reserved
adc_jext_trg0	tim1_trgo
adc_jext_trg1	tim1_trgo2
adc_jext_trg2	tim8_trgo
adc_jext_trg3	tim8_trgo2
adc_jext_trg4	exti14 (from EXTI1)
adc_jext_trg5	Reserved
adc_jext_trg6	tim2_trgo

Table 305. ADC1 interconnection

Signal name	Source/destination
adc_jext_trg7	tim3_trgo
adc_jext_trg8	tim4_trgo
adc_jext_trg9	tim5_trgo
adc_jext_trg10	tim6_trgo
adc_jext_trg11	tim7_trgo
adc_jext_trg12	tim15_trgo
adc_jext_trg13	tim16_oc1
adc_jext_trg14	tim1_oc3
adc_jext_trg15	tim8_oc1
adc_jext_trg16	Reserved
adc_jext_trg17	tim2_oc1
adc_jext_trg18	tim2_oc3
adc_jext_trg19	tim3_oc1
adc_jext_trg20	tim4_oc1
adc_jext_trg21	tim5_oc3
adc_jext_trg22	tim12_oc1
adc_jext_trg23	exti15 (from EXTI2)
adc_jext_trg24	lptim1_ch2
adc_jext_trg25	lptim2_ch2
adc_jext_trg26	lptim3_ch1
adc_jext_trg27	lptim4_ch1
adc_jext_trg28	lptim5_out
adc_jext_trg29	Reserved
adc_jext_trg30	Reserved
adc_jext_trg31	Reserved

Table 306. ADC2 interconnection

Signal name	Source/destination
ADC2 V_{INP}[14]	V_{REFINT} (output voltage from internal reference voltage).
ADC2 V_{INP}[15]	V_{DDCORE} (internal logic core voltage)
ADC2 V_{INP}[16]	V_{BAT}/4 (VBAT pin input voltage divided by 4).
ADC2 V_{INP}[17]	V_{DDCPU} (Internal CPU supply voltage)
ADC2 V_{INP}[19]	V_{REF+}
adc_dat2[15:0]	mdf_adc2_dat[15:0]
adc_ext_trg0	tim1_trgo

Table 306. ADC2 interconnection

Signal name	Source/destination
adc_ext_trg1	tim1_trgo2
adc_ext_trg2	tim8_trgo
adc_ext_trg3	tim8_trgo2
adc_ext_trg4	exti6 (from EXTI1)
adc_ext_trg5	Reserved
adc_ext_trg6	tim2_trgo
adc_ext_trg7	tim3_trgo
adc_ext_trg8	tim4_trgo
adc_ext_trg9	tim5_trgo
adc_ext_trg10	tim6_trgo
adc_ext_trg11	tim7_trgo
adc_ext_trg12	tim15_trgo
adc_ext_trg13	tim17_oc1
adc_ext_trg14	tim1_oc3
adc_ext_trg15	tim8_oc1
adc_ext_trg16	Reserved
adc_ext_trg17	tim2_oc1
adc_ext_trg18	tim2_oc3
adc_ext_trg19	tim3_oc1
adc_ext_trg20	tim4_oc1
adc_ext_trg21	tim5_oc3
adc_ext_trg22	tim12_oc1
adc_ext_trg23	exti2 (from EXTI2)
adc_ext_trg24	lptim1_ch1
adc_ext_trg25	lptim2_ch1
adc_ext_trg26	lptim3_ch1
adc_ext_trg27	lptim4_ch1
adc_ext_trg28	lptim5_out
adc_ext_trg29	Reserved
adc_ext_trg30	Reserved
adc_ext_trg31	Reserved
adc_jext_trg0	tim1_trgo
adc_jext_trg1	tim1_trgo2
adc_jext_trg2	tim8_trgo
adc_jext_trg3	tim8_trgo2

Table 306. ADC2 interconnection

Signal name	Source/destination
adc_jext_trg4	exti7 (from EXTI1)
adc_jext_trg5	Reserved
adc_jext_trg6	tim2_trgo
adc_jext_trg7	tim3_trgo
adc_jext_trg8	tim4_trgo
adc_jext_trg9	tim5_trgo
adc_jext_trg10	tim6_trgo
adc_jext_trg11	tim7_trgo
adc_jext_trg12	tim15_trgo
adc_jext_trg13	tim1_oc3
adc_jext_trg14	tim1_oc4
adc_jext_trg15	tim8_oc2
adc_jext_trg16	tim8_oc4
adc_jext_trg17	Reserved
adc_jext_trg18	tim4_oc3
adc_jext_trg19	tim4_oc4
adc_jext_trg20	tim5_oc1
adc_jext_trg21	tim12_oc2
adc_jext_trg22	tim15_oc1
adc_jext_trg23	exti3 (from EXTI2)
adc_jext_trg24	lptim1_ch2
adc_jext_trg25	lptim2_ch2
adc_jext_trg26	lptim3_ch1
adc_jext_trg27	lptim4_ch1
adc_jext_trg28	lptim5_out
adc_jext_trg29	Reserved
adc_jext_trg30	Reserved
adc_jext_trg31	Reserved

36.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 255: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

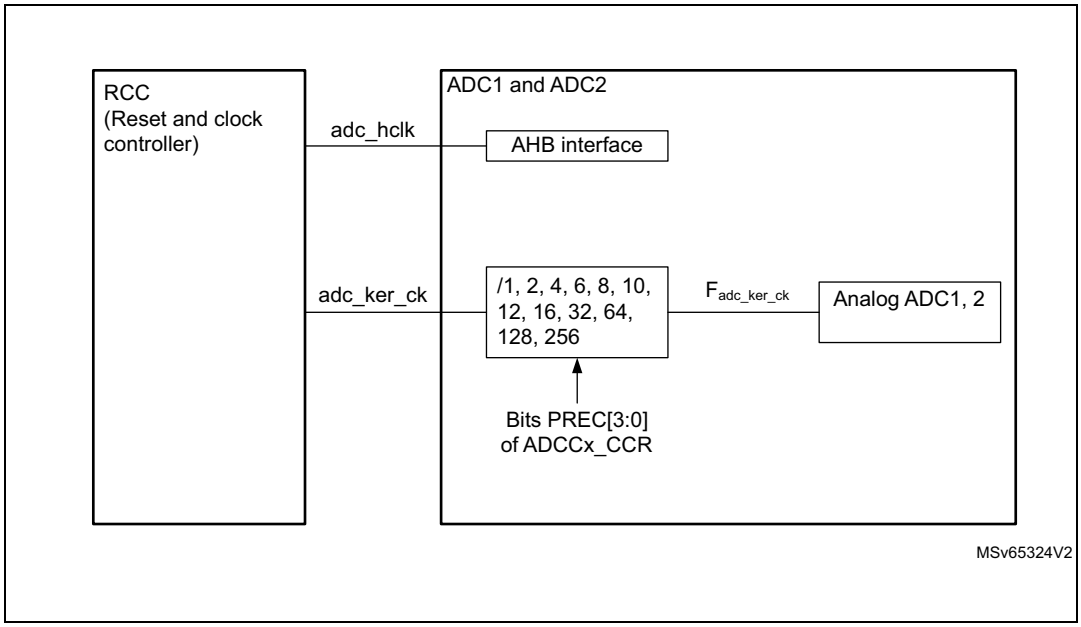
1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can be divided by a 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, or 256 ratio, using the prescaler configured through the PRESC[3:0] bits of the ADCC_CCR register.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 255. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio, for all ADC interfaces.

36.4.4 ADC connectivity

ADC inputs are connected to external channels as well as internal sources as described below.

Figure 256. ADC1 connectivity

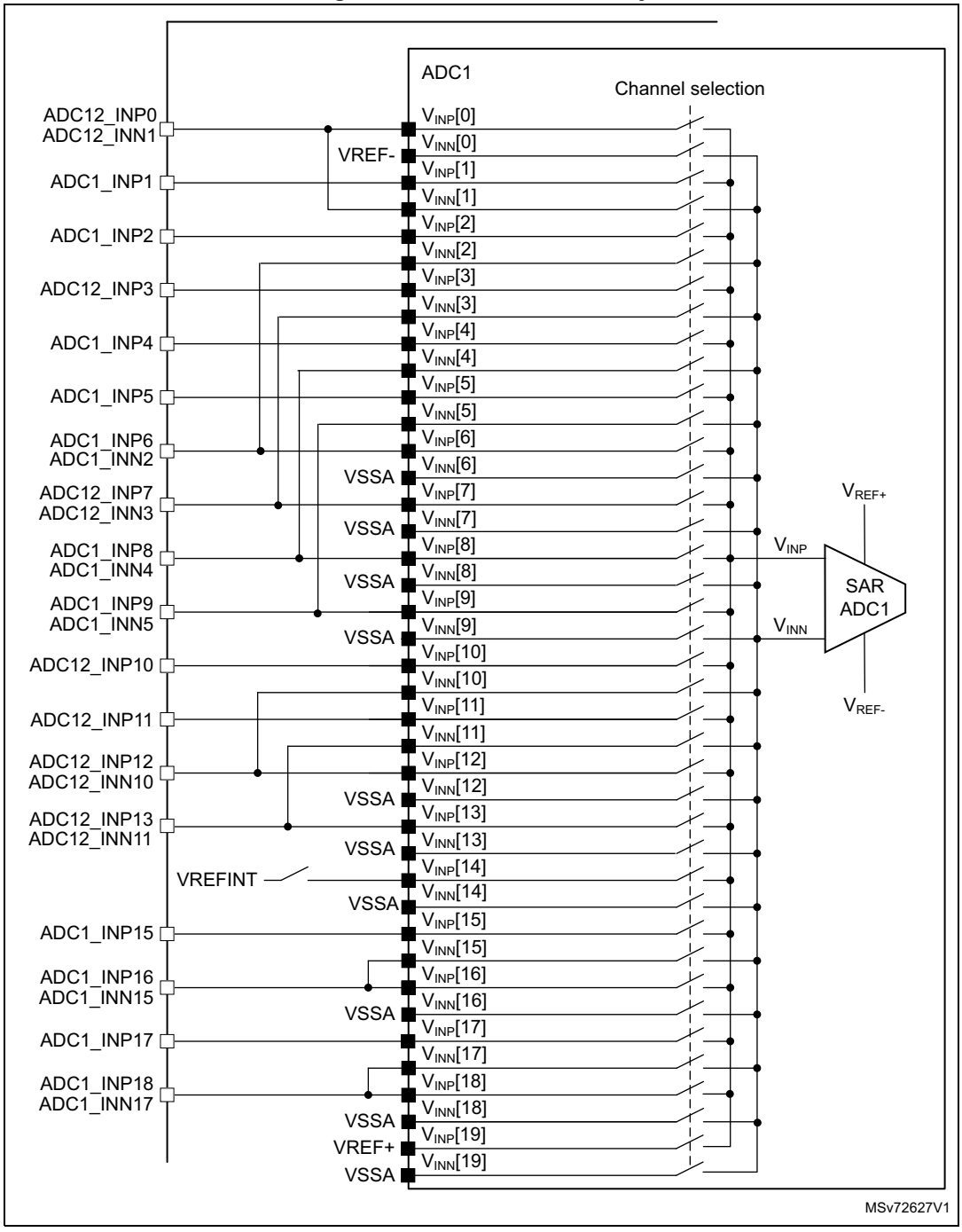
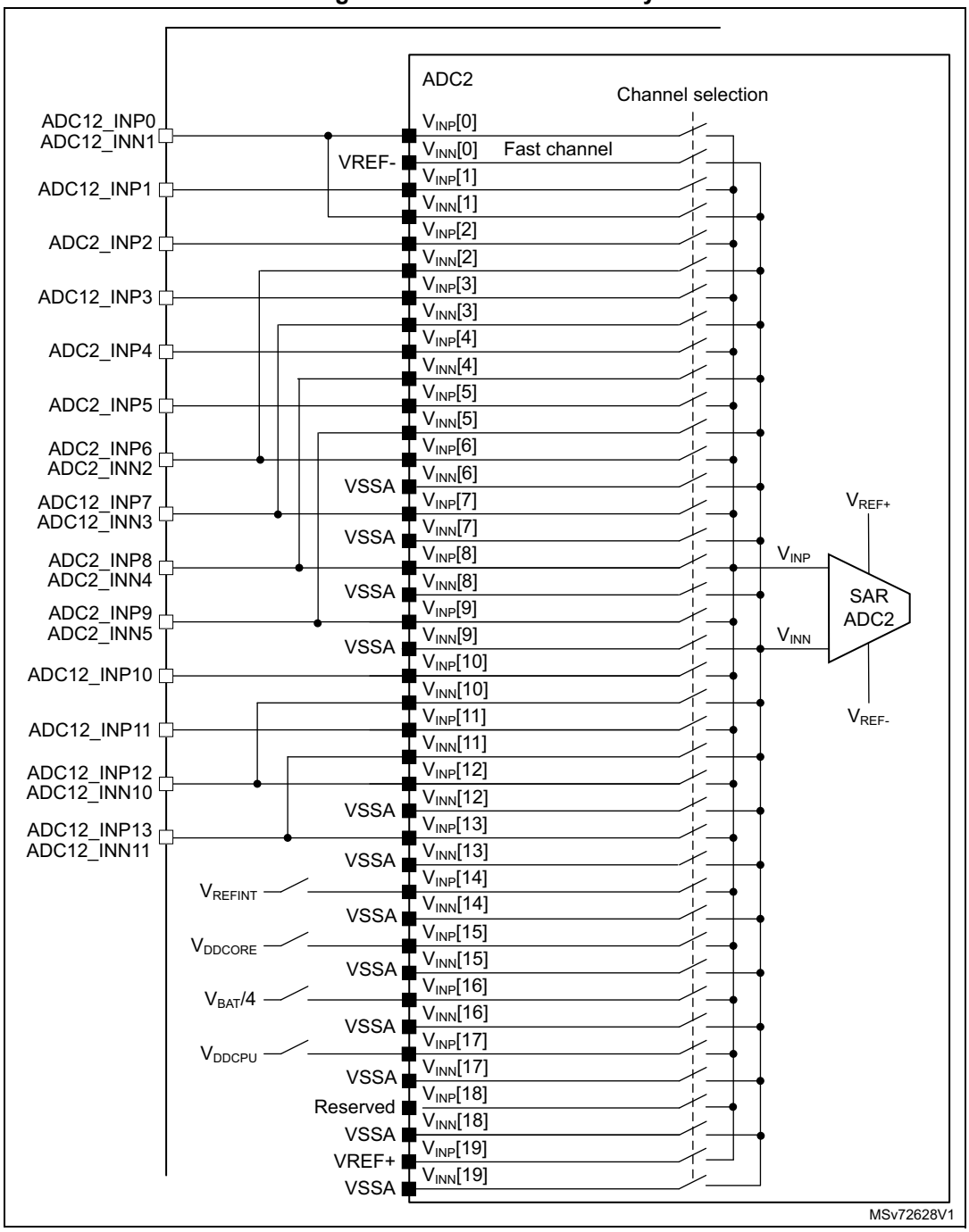


Figure 257. ADC2 connectivity



36.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states, at the exception of ADC_ISR and ADC_AWDyLTR/HTR registers for which wait cycles are added for write operations.

The AHB slave interface does not support split/retry requests, and it can generate AHB error in case of wrong HSIZE.

36.4.6 ADC Deep-power-down mode (DEEPPWD)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of DEEPPWD bit is 1 in the ADC_CR register).

To activate the ADC, first exit the Deep-power-down mode by clearing the DEEPPWD bit.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of the ADC_CR register.

Power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

36.4.7 Single-ended and differential input channels

ADC channels can be configured to be either single-ended inputs or differential inputs by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be programmed when no ADC conversion is ongoing. Note that the DIFSEL[i] bits corresponding to single-ended channels must be programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ and V_{REF-}.

In differential input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

In differential mode, the output data are unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When the ADC is configured in differential mode, the input signals are supposed to be differential (the common-mode voltage must be fixed at $V_{REF+}/2$ voltage).

Internal channels are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 36.4.4: ADC connectivity*.

Caution: When configuring channel "*i*" in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended or differential mode, and must never be configured to be converted. Some channels are shared between ADCs: this can make the channel on the other ADCs unusable. The only exception is interleaved mode for ADC master and the slave.

36.4.8 Calibration (ADCAL, CALADDOS, ADC_CALFACT)

A software calibration is required before launching the application. During the calibration sequence, the ADC calculates a 9-bit calibration factor, which is then internally applied to the ADC. Each ADC must follow the calibration procedure to calibrate the zero offset.

The calibration removes the offset error that may vary from chip to chip due to process or bandgap variation. It is a prerequisite to any ADC operation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Clear ADCALDIF before launching a calibration that is applied for single-ended input conversions.
- Set ADCALDIF before launching a calibration that is applied for differential input conversions.

The calibration needs to be run by combinations of hardware and software. When the calibration procedure is complete, the calibration factor must be written to the CALFACT_S[8:0] bits for the single-ended calibration factor and CALFACT_D[8:0] for the differential calibration factor.

The calibration factor can be written only if the ADC is enabled and no conversion is ongoing (ADEN = 1, ADSTART = 0 and JADSTART = 0). It is recommended to calibrate again the ADC if V_{REF+} voltage changes of more than 10% (see *Section 36.4.32: Monitoring the internal voltage reference*).

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADEN = 1 and wait until the ADRDY bit is set.
2. Set ADCAL and ensure CALADDOS = 0.
3. Select the calibration input mode by clearing ADCALDIF (single-ended input).
4. Set the ADSTART bit in the ADC_CR register.
5. Wait until the ADSTART bit is cleared or the EOC flag is set.
6. Read the ADC_DR register, then copy the converted data to the memory.
7. Repeat from step 4 several times (for example eight times).
8. Average the data stored in memory by dividing the accumulated data by the number of the conversions
9. If the averaged data is zero, set CALADDOS. Repeat all steps from step 4.
10. Store the averaged data to CALFACT_S[8:0].
11. Select the calibration input mode by setting ADCALDIF (differential input).
12. Keep the same CALADDOS setting as the one obtained during the single-end calibration.
13. Repeat steps 4 to 8.
14. Subtract 0x7FF from the averaged data. If the result is positive, store it in the CALFACT_D[8:0] bitfield. If it is negative, set CALADDOS, then repeat steps from 4 to 8.
15. Subtract again 0x7FF from the new averaged data. The resulting value is positive. Store it in CALFACT_D[8:0].
16. CALADDOS is now set, so clear ADCALDIF, and repeat steps 4 to 8.
17. Store the averaged data in CALFACT_S[8:0].
18. Clear ADCAL bit.

Software procedure to reinject a calibration factor into the ADC

1. Ensure ADSTART = 0 and JADSTART = 0 (no conversion ongoing).
2. Set ADCAL.
3. Program CALADDOS, CALFACT_S and CALFACT_D with the new calibration factors.
4. Clear ADCAL.

Note: CALFACT_S and CALFACT_D limit the maximum value of the ADC digital output. For example, in single-ended mode, if CALFACT_S = 100, the maximum ADC data output is 4095 - 100 = 3995. When the CALADDOS bit is set, an analog positive offset of about 256 is internally added to the ADC input to allow the calibration of negative offset devices. The resulting calibration factor is approximately 256. The maximum ADC output is around 4095 - 256 = 3839.

Converting single-ended and differential analog inputs with a single ADC

If the ADC has to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1.

36.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First, follow the procedure described in *Section 36.4.6: ADC Deep-power-down mode (DEEPPWD)*).

Once DEEPPWD is cleared, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 258*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 36.4.20: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if regular triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

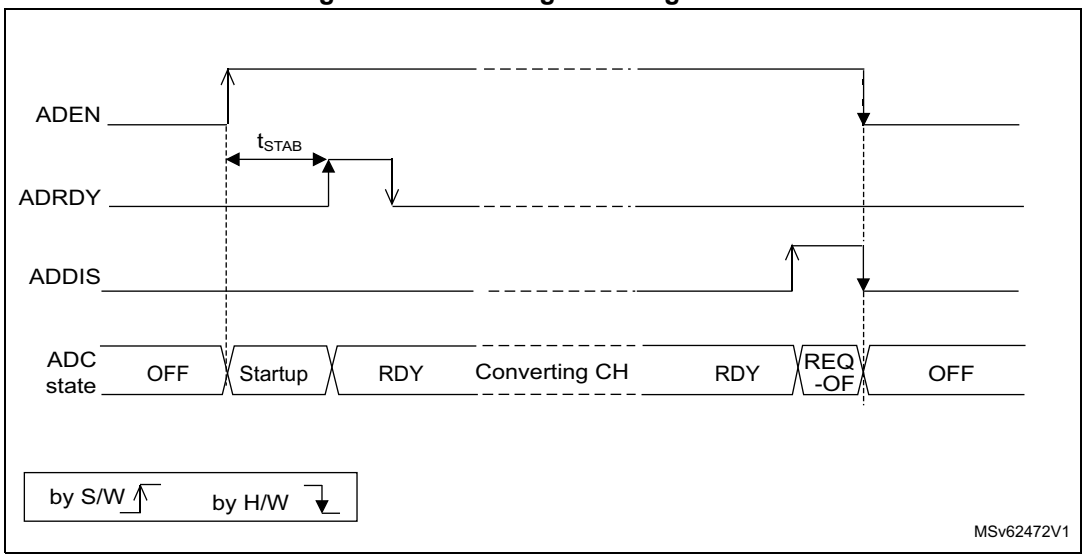
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 258. Enabling/disabling the ADC



36.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy register thresholds can be modified when an analog-to-digital conversion is ongoing (refer to *Section 36.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for details).
- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC.

Note: Not all forbidden write accesses to ADC control bits are protected by hardware. In some cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

36.4.11 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 18 analog inputs coming from GPIO pads (ADCx_INN/INP[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 5 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - V_{BAT} monitoring channel ($V_{BAT}/4$)
  - V_{REF+} positive reference voltage
  - V_{DDCORE} voltage
  - V_{DDCPU} voltage

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, VBATEN in the ADCC_CCR registers, or by enabling the corresponding bit in the *ADC option register (ADC_OR)*.

Refer to *ADC interconnection* tables in *Section 36.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INN/INP3, ADCx_INN/INP8, ADCx_INN/INP2, ADCx_INN/INP2, ADCx_INN/INP0, ADCx_INN/INP2, and ADCx_INN/INP2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the JLEN[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 36.4.19: Stopping an ongoing conversion (ADSTP, JADSTP)*).

36.4.12 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{INP} of each ADC input. For example, if the ADC converts ADCx_INN/INP1, PCSEL1 bit must also be set in ADC_PCSEL register. If ADCx_INN/INP1 is configured as a differential channel, both PCSEL1 and PCSEL0 must be set, which corresponds to ADC1_INN1 and ADC2_INN1.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

36.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 6.5 ADC clock cycles
- SMP = 011: 11.5 ADC clock cycles
- SMP = 100: 23.5 ADC clock cycles
- SMP = 101: 46.5 ADC clock cycles
- SMP = 110: 246.5 ADC clock cycles
- SMP = 111: 1499.5 ADC clock cycles

The total conversion time is calculated as follows (12-bit mode):

$$T_{CONV} = \text{Sampling time} + 13.5 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 30 MHz and a sampling time of 1.5 ADC clock cycles:

$$T_{CONV} = (1.5 + 13.5) \times \text{ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPRx registers. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

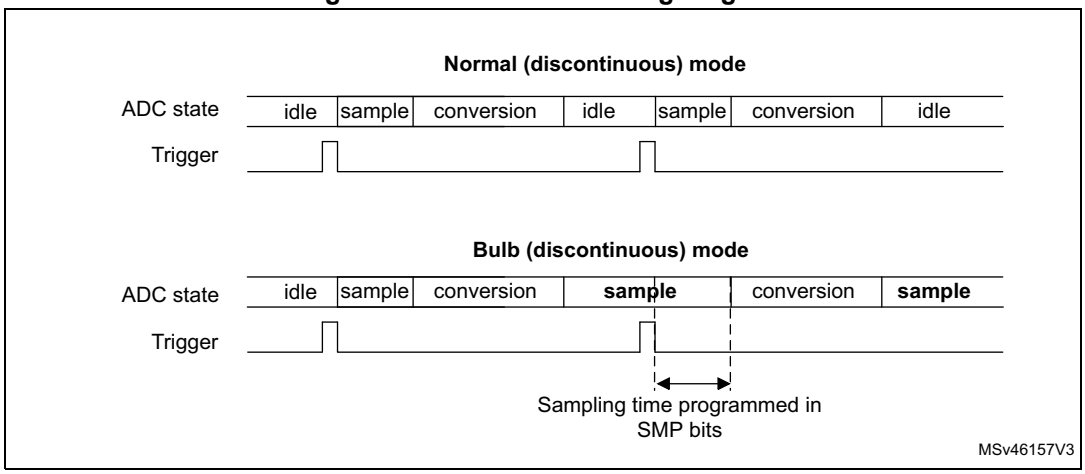
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

Bulb mode is supported exclusively in regular conversion; dual mode is not supported.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 259. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and only regular conversion is supported.

36.4.14 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

36.4.15 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

36.4.16 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN and CONT = 0 bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the LEN[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.
  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels with DISCNUM = 0.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JLEN[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

36.4.17 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

When the hardware trigger is enabled, ADSTART and JADSTART also indicate whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

In the case of software triggering, ADSTP and/or JADSTP must be set before ADC reconfiguration.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion)
- In discontinuous mode (CONT = 0, DISCEN = 1, DISCNUM= x): at any of subgroup sequence
- In all cases (CONT and EXTEN don't care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion)
- In discontinuous mode (JDISCEN = 1): at the end of any conversion
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

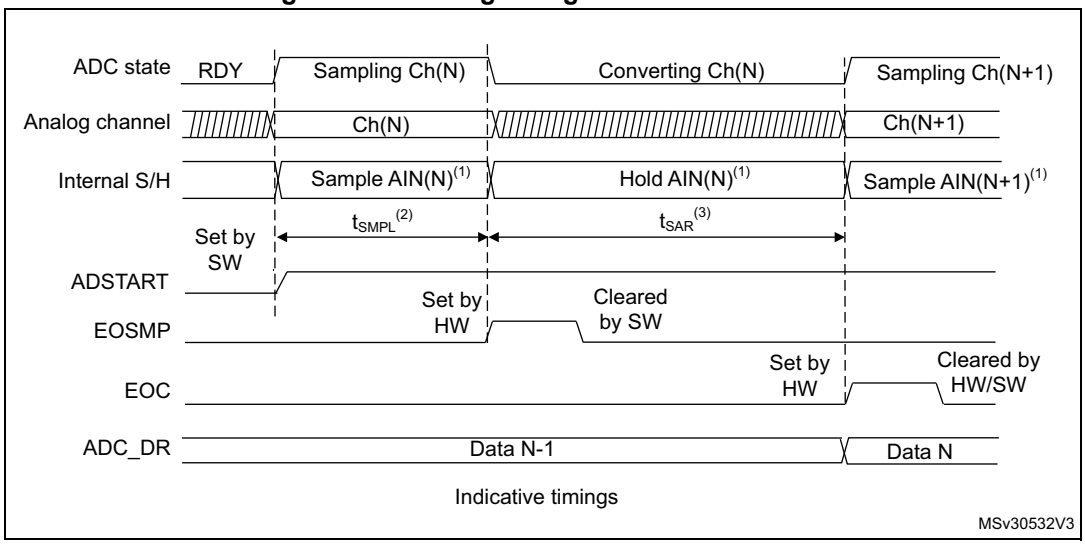
36.4.18 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [1.5\ _{|min} + 13.5\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 50\ ns\ _{|min} + 450\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 30\ MHz$)

Figure 260. Analog-to-digital conversion time



1. AIN represents the input voltage captured by the internal Sample and Hold (S/H) capacitor.
2. t_{SMPL} depends on SMP[2:0].
3. t_{SAR} depends on RES[2:0].

Note: The bulb and the sampling time control trigger modes are not described in Figure 260: Analog-to-digital conversion time.

36.4.19 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 261. Stopping ongoing regular conversions

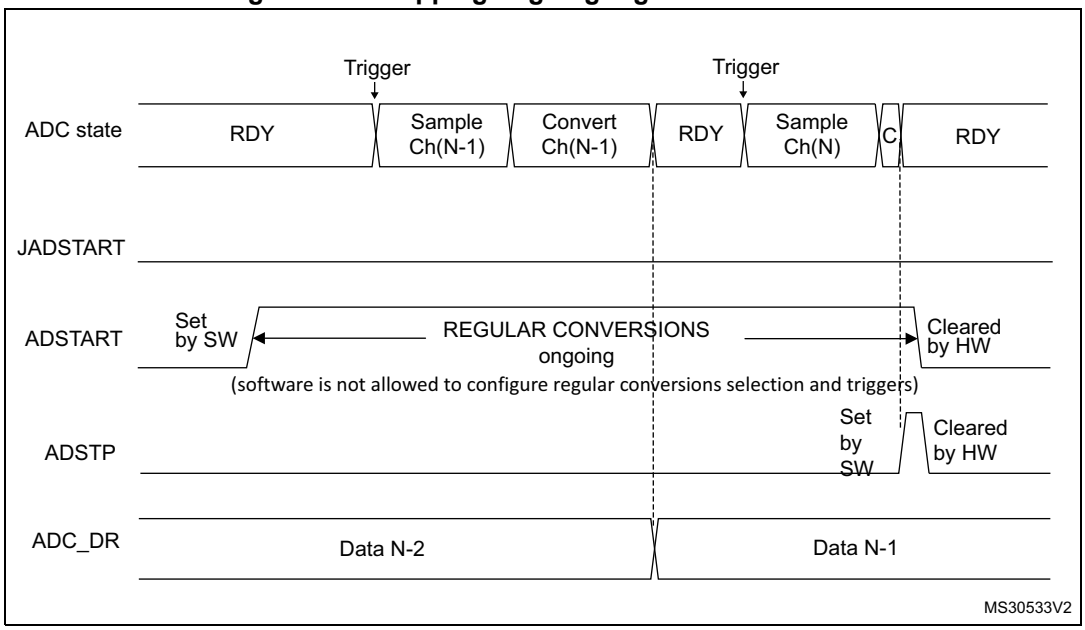
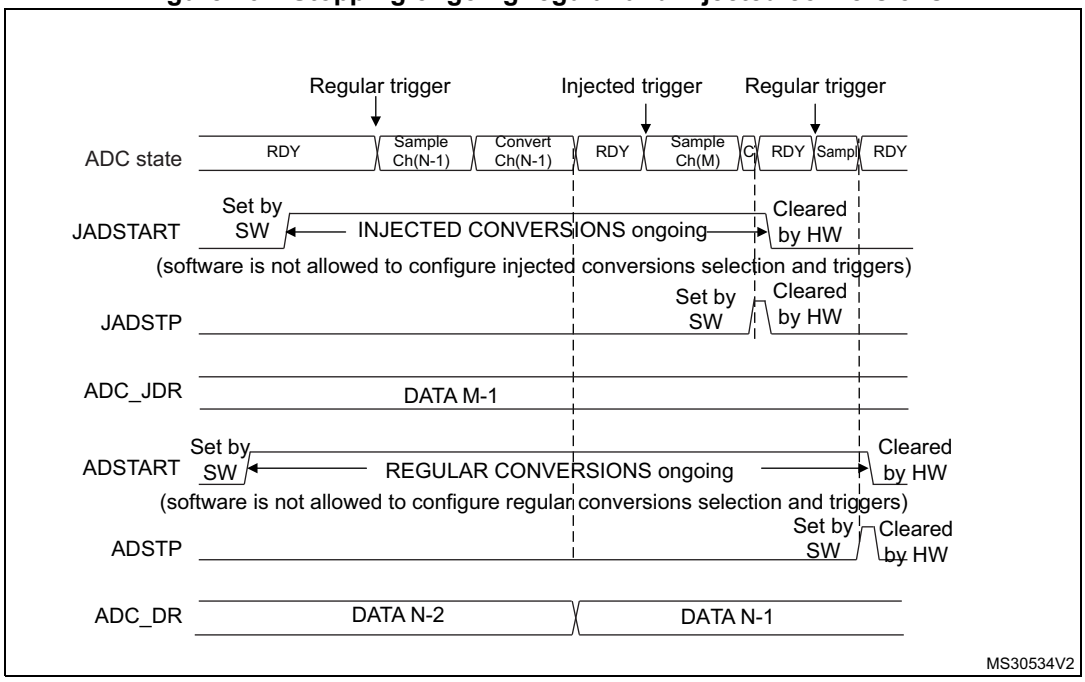


Figure 262. Stopping ongoing regular and injected conversions



36.4.20 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 307 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 307. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in SMPTRIG mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 308. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Refer to *ADC interconnection* tables in *Section 36.4.2: ADC pins and internal signals* for the list of the external triggers.

36.4.21 Injected channel management

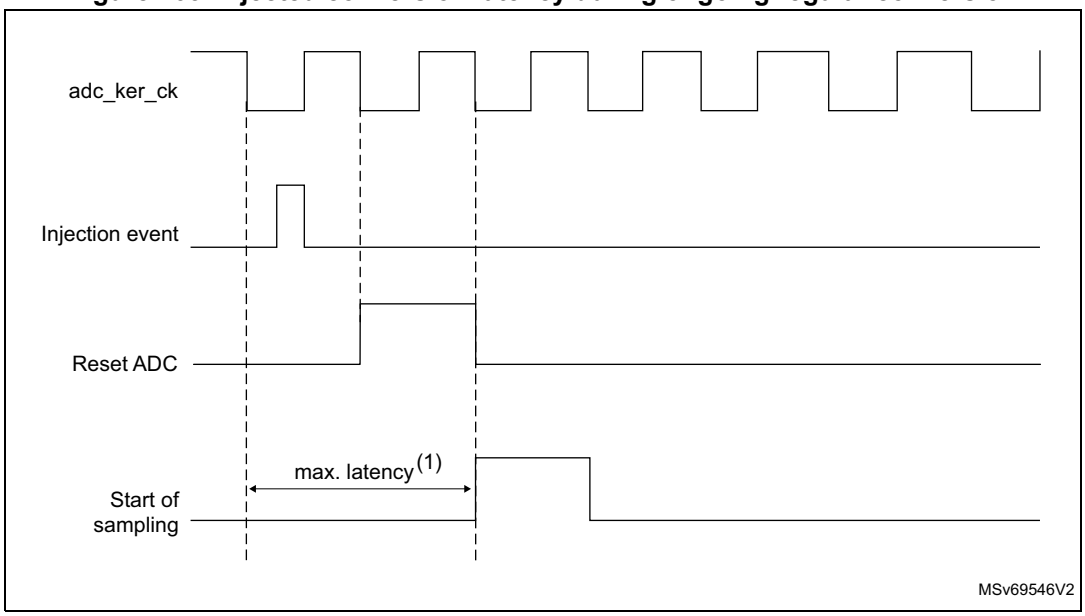
Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 263* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence.

Figure 263. Injected conversion latency during ongoing regular conversion



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

Note: Autoinjection mode is compatible with discontinuous mode.

36.4.22 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 268*, *Figure 269*, *Figure 270* and *Figure 271* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 309*.

Table 309. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 1.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 30 MHz
12	13.5 ADC clock cycles	450 ns	15 ADC clock cycles	500.0 ns
10	11.5 ADC clock cycles	383.33 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	10 ADC clock cycles	333.33 ns
6	6.5 ADC clock cycles	216.67 ns	8 ADC clock cycles	266.66 ns

36.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

36.4.24 End of conversion sequence (EOS, JEOS)

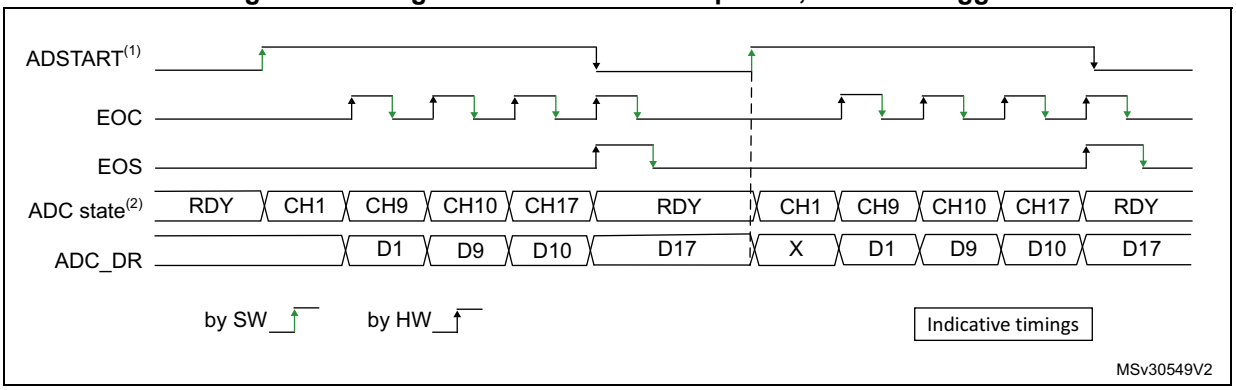
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

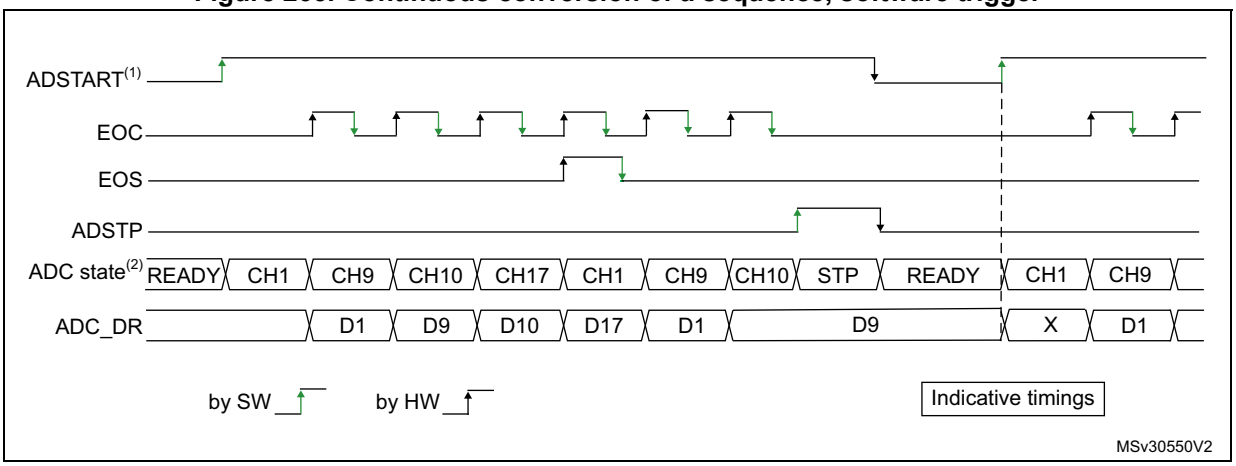
36.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 264. Single conversions of a sequence, software trigger



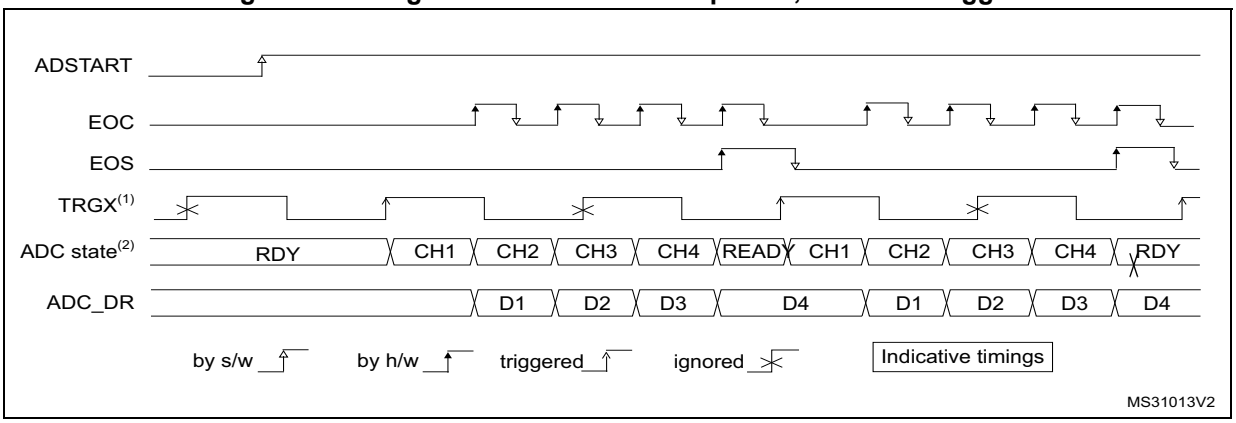
1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 265. Continuous conversion of a sequence, software trigger



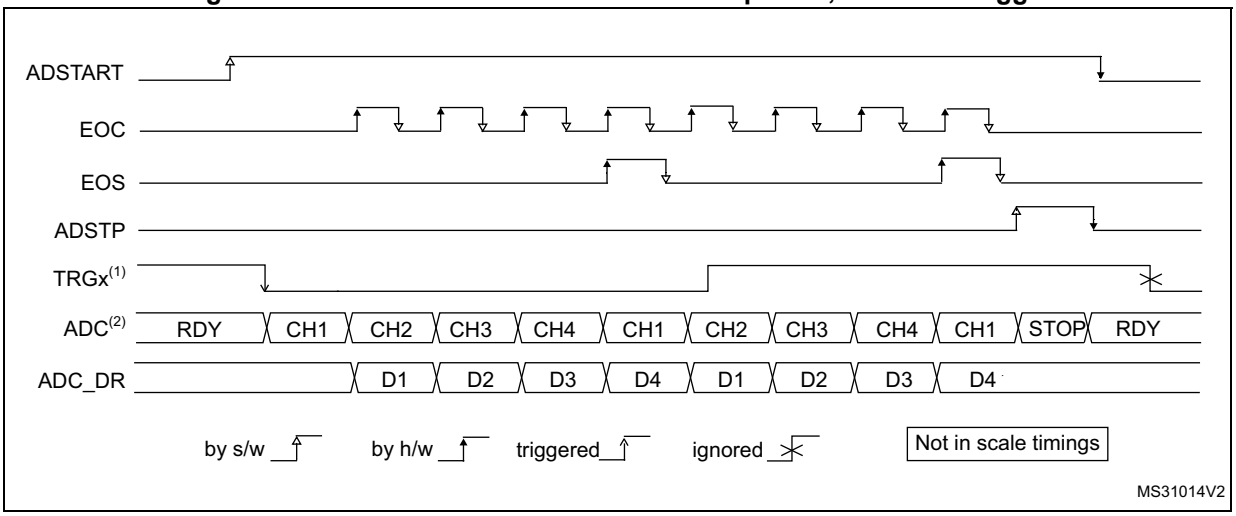
1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 266. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 267. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

36.4.26 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS[a], JLSHIFT[a], USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO (when OVRMOD = 0). Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0], LSHIFT[3:0], JOVSS[3:0], and JLSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 268*, *Figure 269*, *Figure 270*, and *Figure 271* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

JOVSS[3:0] and JLSHIFT[3:0] is reserved for injected conversions. Their usage is the same as OVSS[3:0] and LSHIFT[3:0], unless otherwise specifically described.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: If the OFFSET channel is selected but the offset value is 0x0000, this offset programming is ignored. If the same OFFSET channel is programmed in different registers multiple times, the first programmed register takes priority.

Table 315 describes how the computation is performed for all the possible ADC resolutions and offset settings.

a. Only for products that include this bit.

Table 310. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 268, *Figure 269*, *Figure 270* and *Figure 271* show alignments for signed and unsigned data. JOVSS and JLSHIFT are not described here, but they behave the same as OVSS and LSHIFT.

Figure 268. Right alignment (offset disabled, unsigned value)

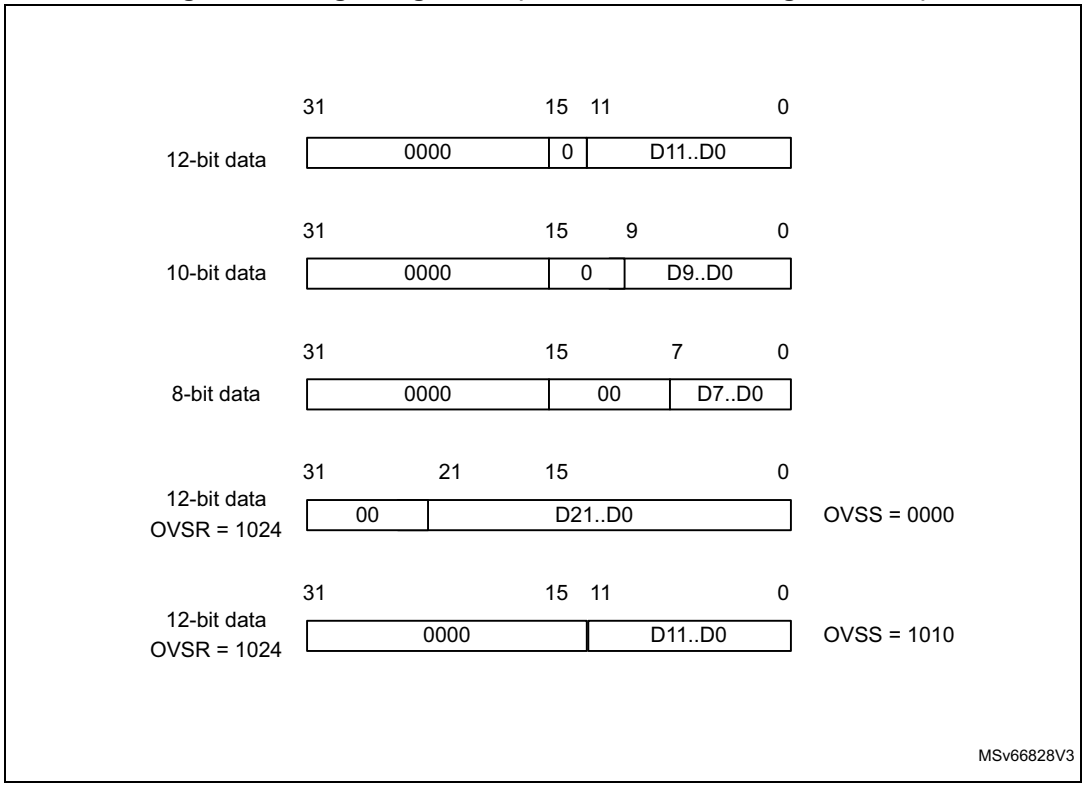


Figure 269. Right alignment (offset enabled, signed value)

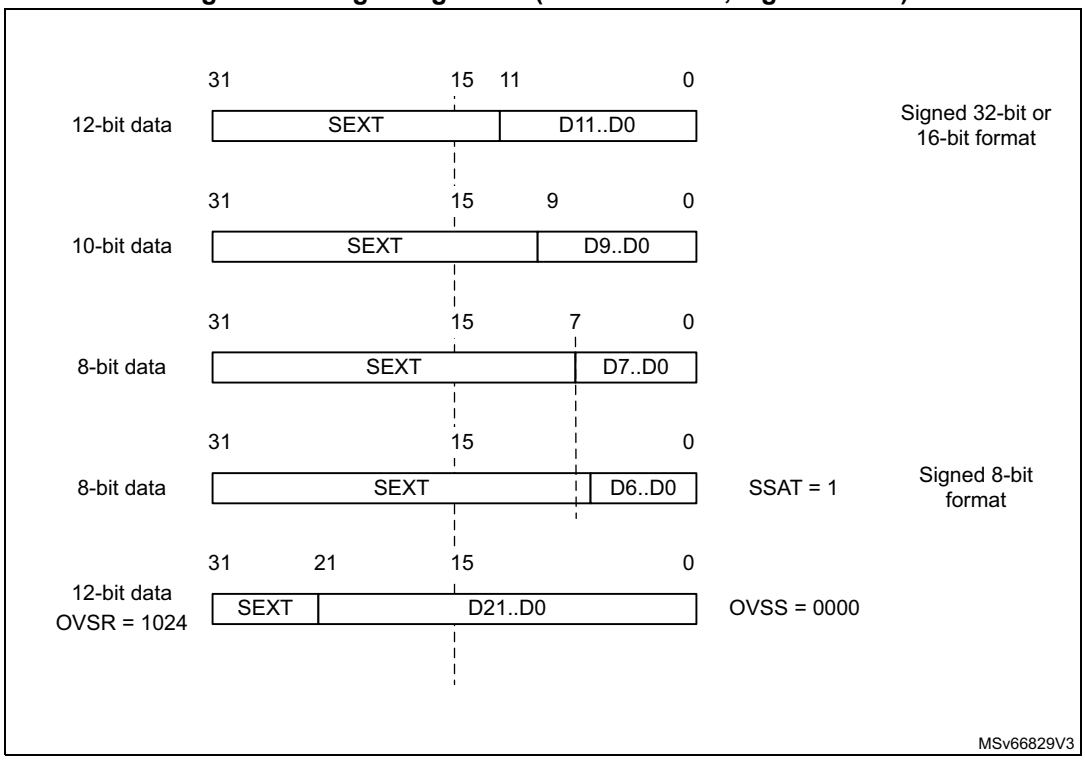


Figure 270. Left alignment (offset disabled, unsigned value)

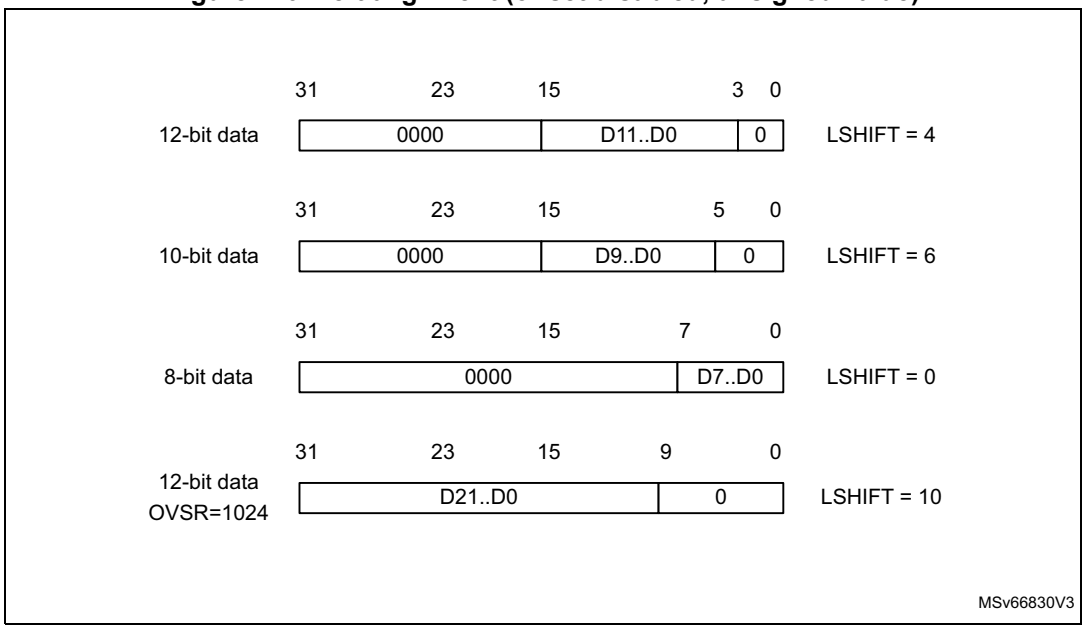
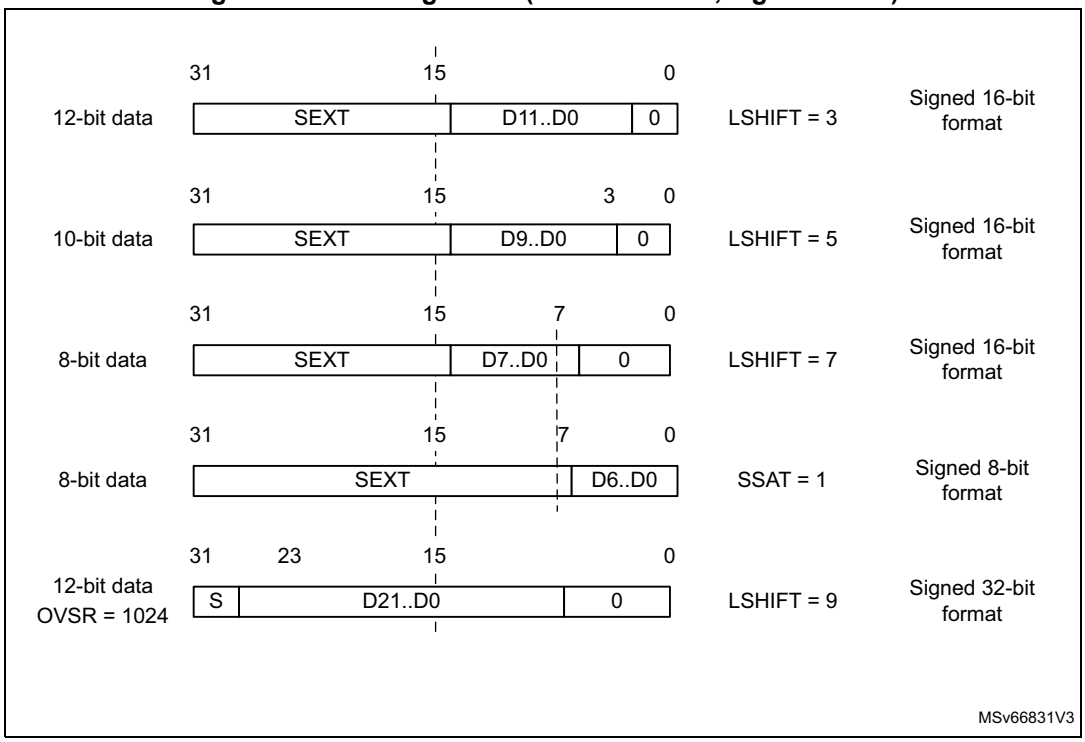


Figure 271. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 311 shows the sign-extended data format corresponding to different resolutions.

Table 311.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 311.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 312 provides numerical examples for four different offset values.

Table 312. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0)

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB - 1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC data latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data when OVRMOD = 0.

The OVR flag is set when a new conversion completes while ADC_DR as single register or FIFO output is full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded. The FIFO can be emptied through ADC_DR read access by the CPU while the OVR flag is set.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode nor MDF output mode cannot be supported.

Figure 272. Example of overrun (OVRMOD = 0)

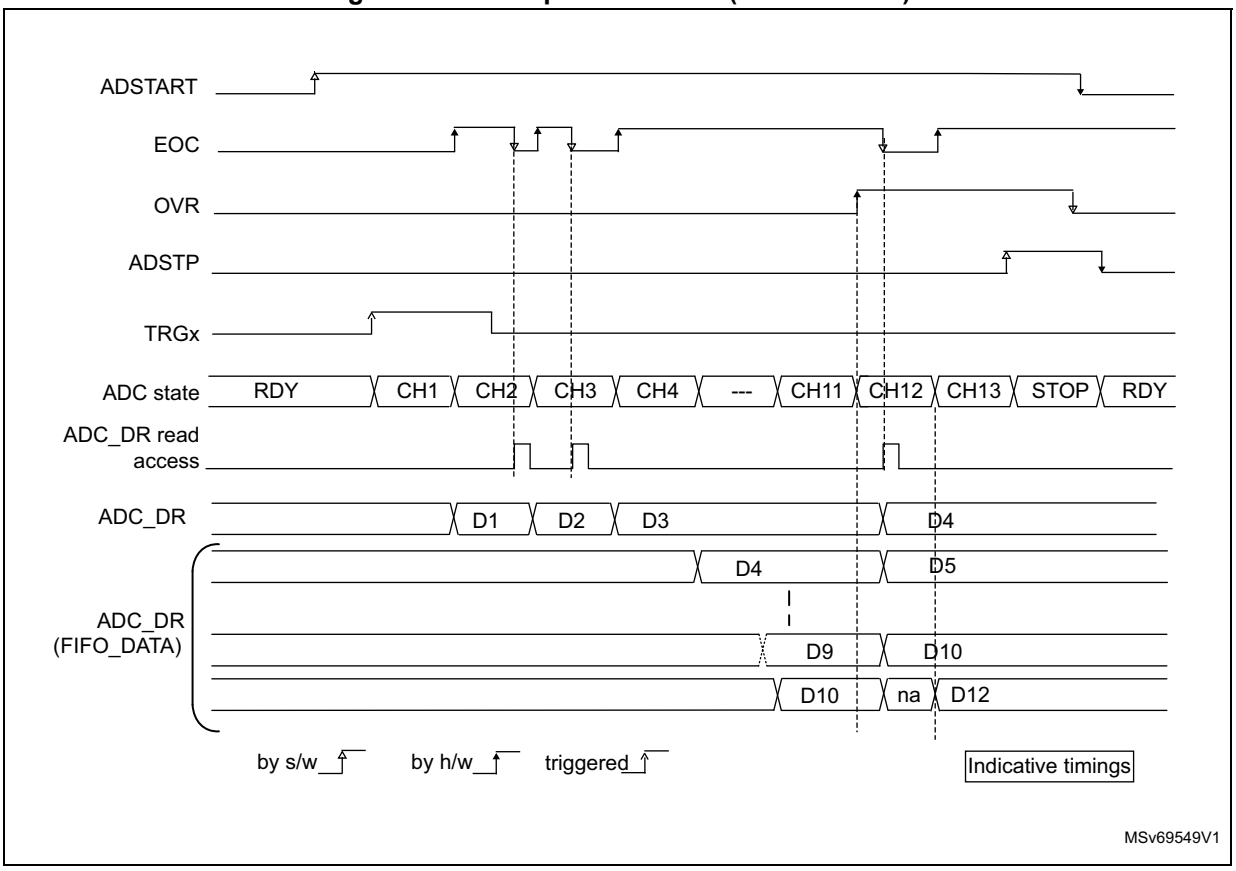
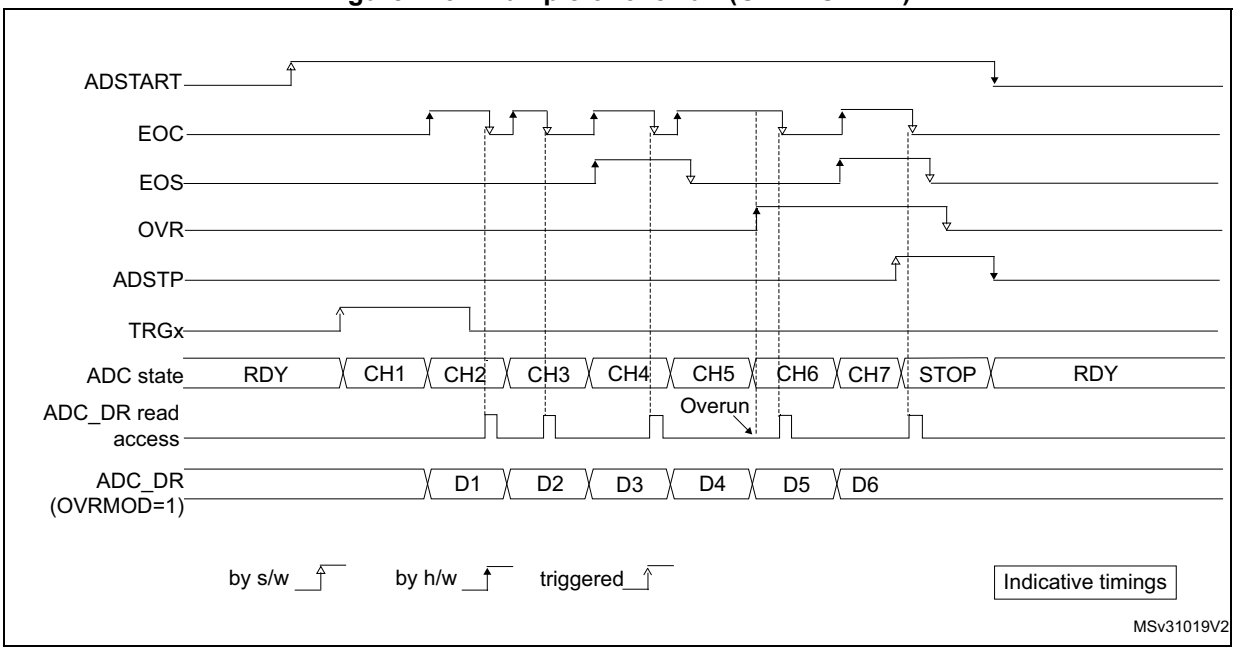


Figure 273. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid. (OVRMOD = 0).

When OVRMOD = 1, even if DMA does not transfer the data immediately, new data overwrite ADC_DR, causing DMA to potentially skip data. As a result, some conversion results may be missed in the data transferred to RAM.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

DMA can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register.

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA when the number of transfers is unknown.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- All ongoing conversions are aborted with the partial result discarded.
- The scan sequence is stopped and reset.
- The DMA and ADC are stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA single-transfer or burst mode with FIFO (OVRMOD = 0)

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

36.4.27 Managing conversions using the MDF

The ADC conversion results can be transferred directly to the MDF.

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers the 16 least significant bits of the regular data register to the MDF through adcx_dat[15:0] bus, which in turn resets the EOC flag once the transfer is effective.

The data format must be in 16-bit signed format:

ADC_DR[31:16] = Don’t care

ADC_DR[15] = Sign

ADC_DR[14:0] = Data

Any value above the 16-bit signed format is truncated.

To obtain a 16-bit signed format, the OFFSET function needs to be activated.

In this mode, OVRMOD = 1 is not supported.

36.4.28 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit in ADC_CFGR1. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- The ADC_DR register is read or the EOC bit is cleared (see *Figure 274*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data and prevent overrun errors.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 275*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

A hardware trigger event occurring during this delay is ignored. .

No delay is inserted between regular and injected conversions. The autodelay for regular and injected modes is managed separately:

Caution: The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 278*).

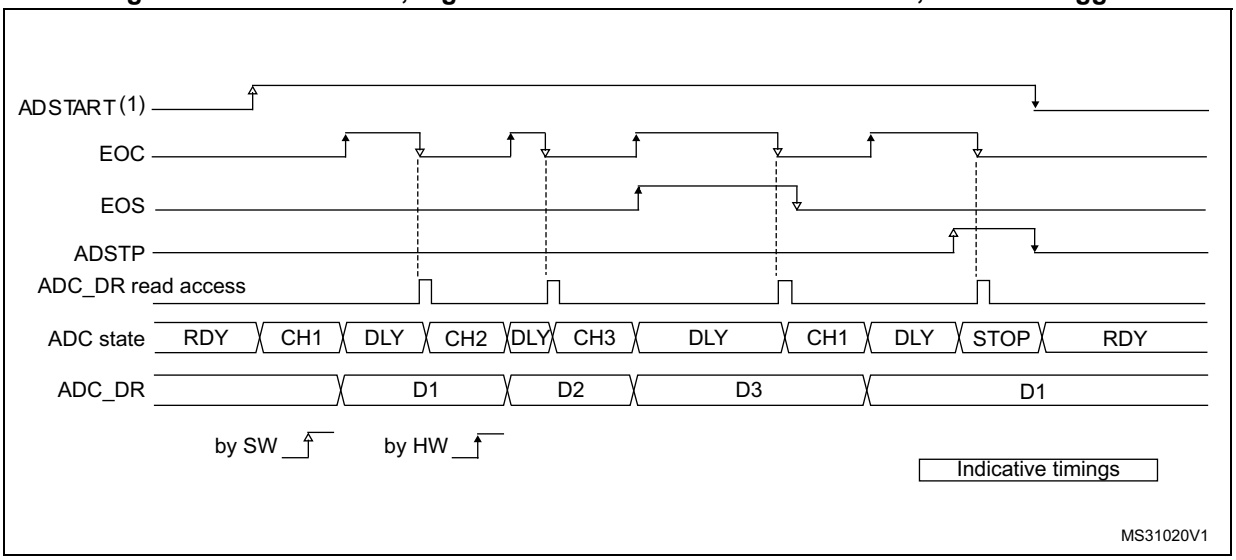
In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In autodelay mode with JAUTO = 0, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

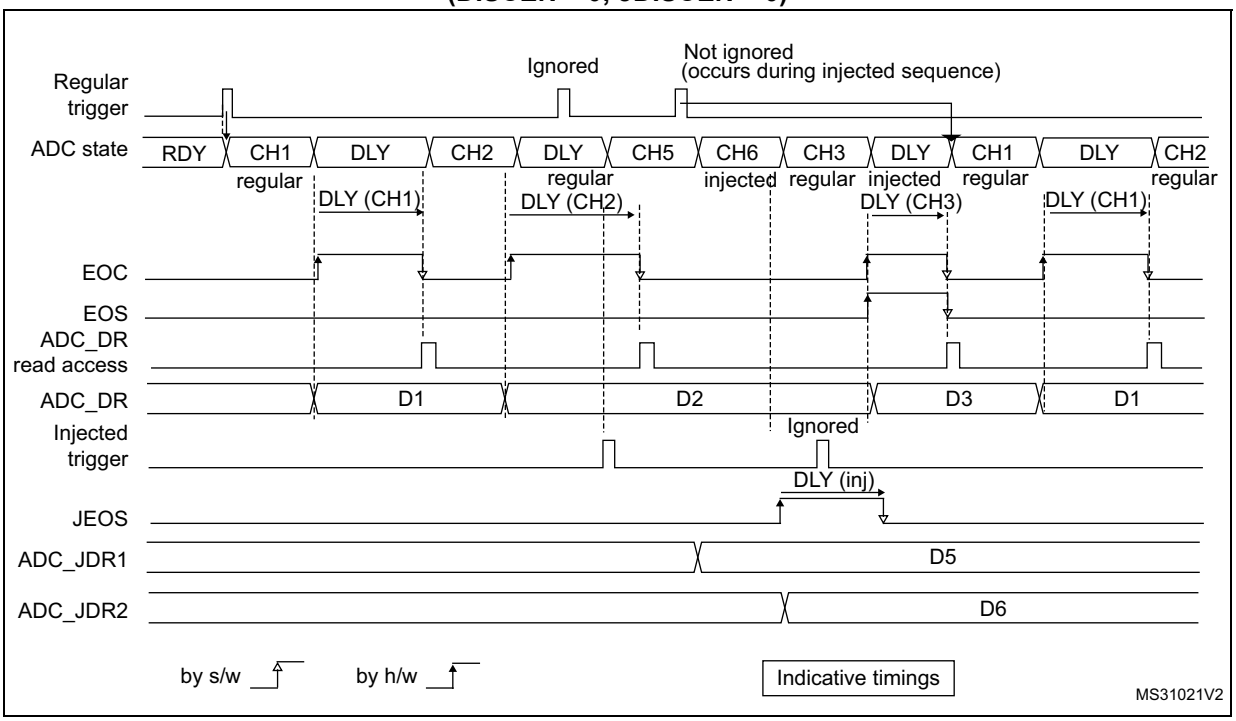
Note: AUTDLY mode is not supported with SMPTRIG or BULB mode.

Figure 274. AUTDLY = 1, regular conversion in continuous mode, software trigger



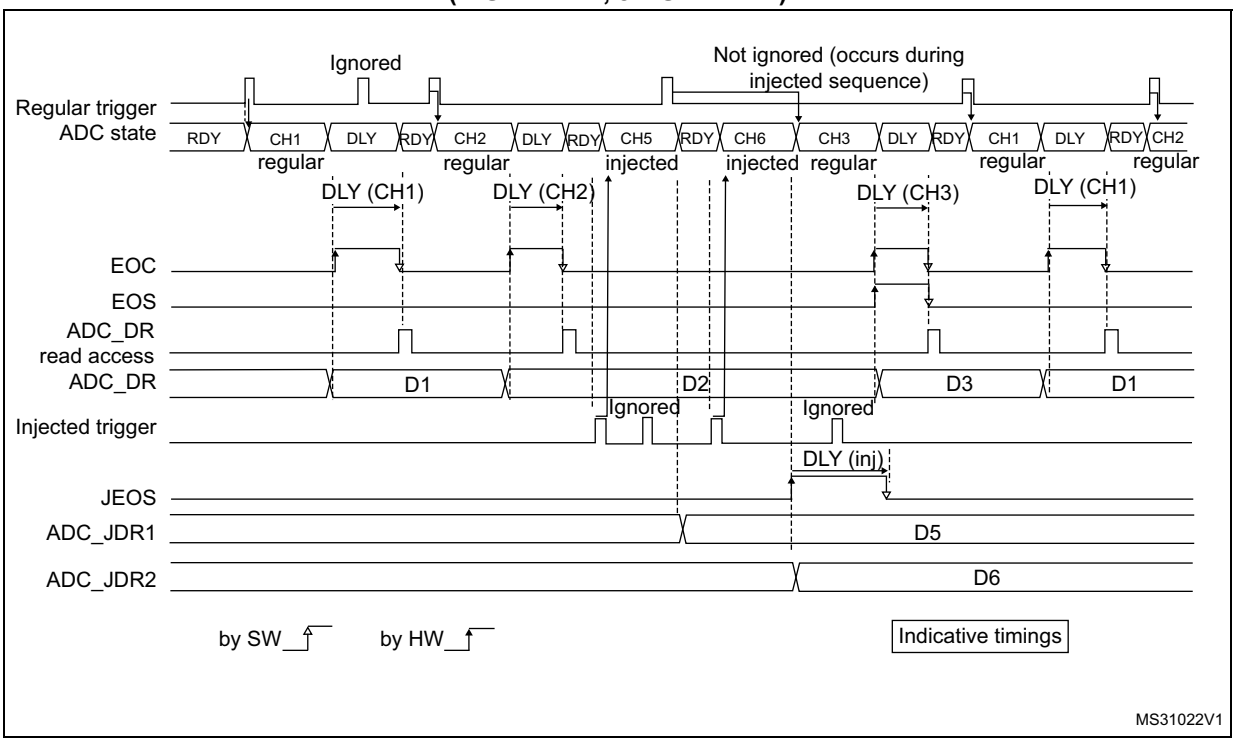
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 275. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



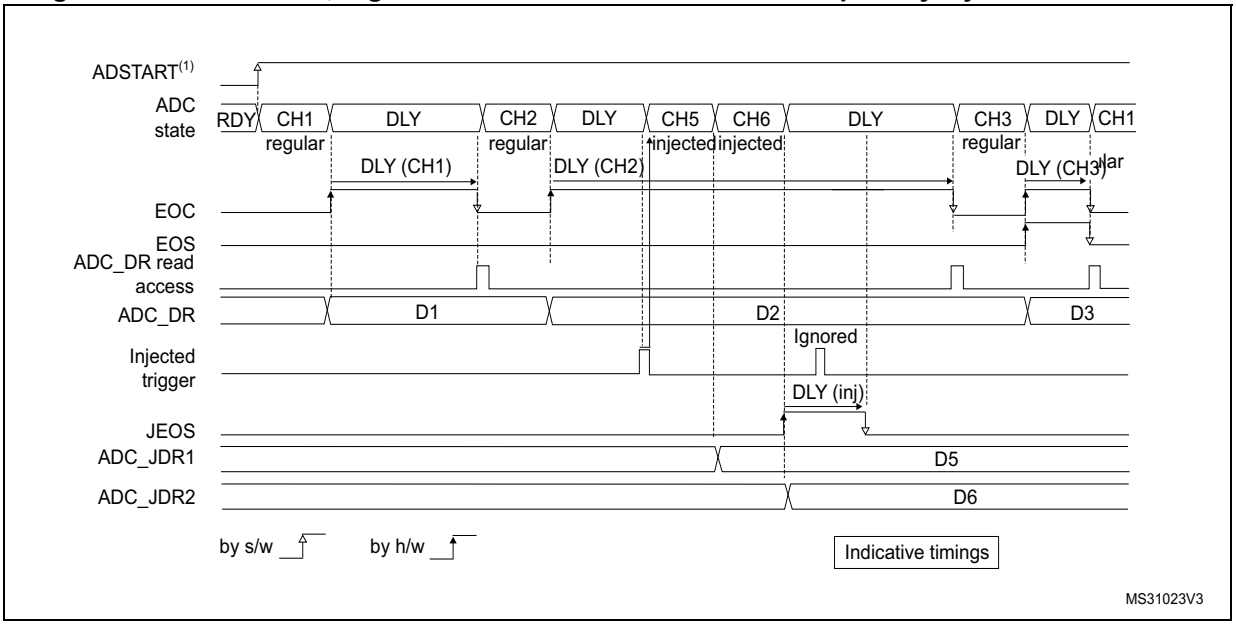
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 276. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



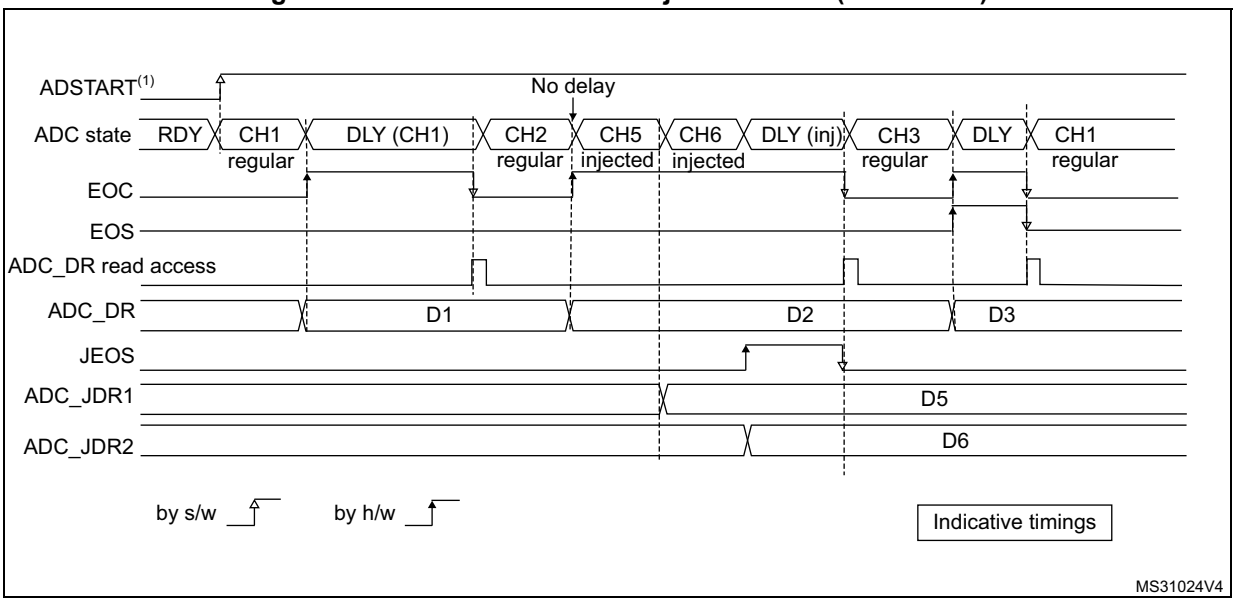
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5, 6.

Figure 277. AUTDLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 278. AUTDLY = 1 in autoinjection mode (JAUTO = 1)



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6.

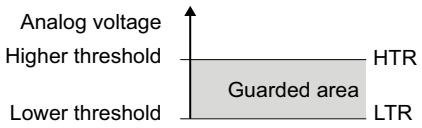
36.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

Table 313. Analog window watchdog features

Feature	AWD1	AWD2	AWD3
Enable control	AWD1EN or JAWD1EN	ADC_AWD2CR	ADC_AWD3CR
Channel selection	AWD1CH[4:0]/AWD1SGL	ADC_AWD2CR	ADC_AWD3CR
Channel selection limitation	One or all channels	Any number of channels	
	Regular or injected	-	-
Threshold control	ADC_AWD1LTR / ADC_AWD1HTR	ADC_AWD2LTR / ADC_AWD2HTR	ADC_AWD3LTR / ADC_AWD3HTR
Filter configuration	AWDFILT[2:0] in ADC_AWDyHTR		
Interrupt	AWDy in ADC_ISR		

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 279. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN or JAWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 314 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 314. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 315 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 315. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 280. ADC_AWDy_OUT signal generation (on all regular channels)

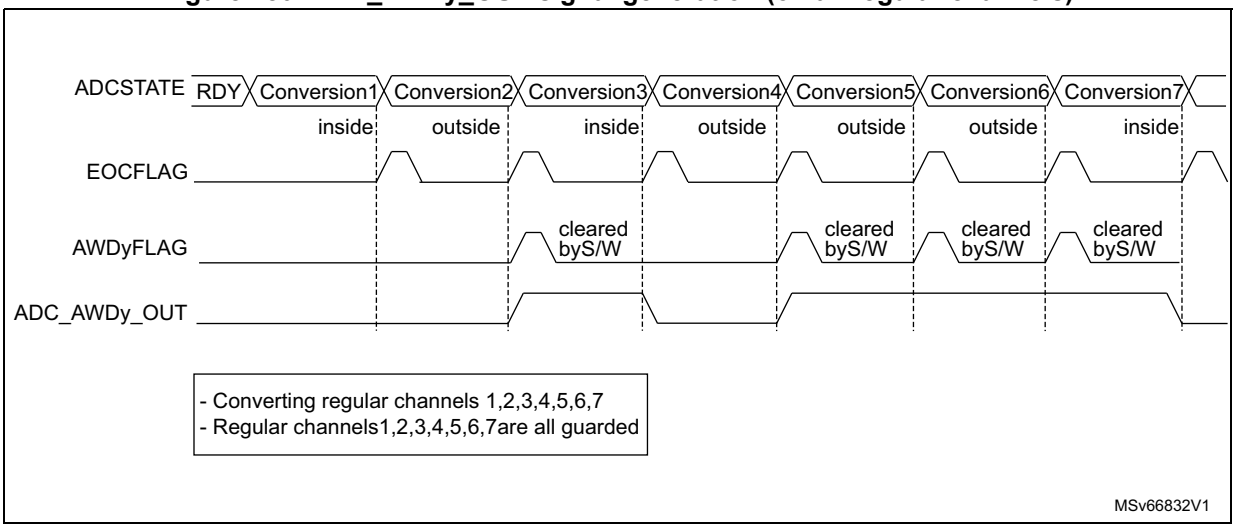


Figure 281. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

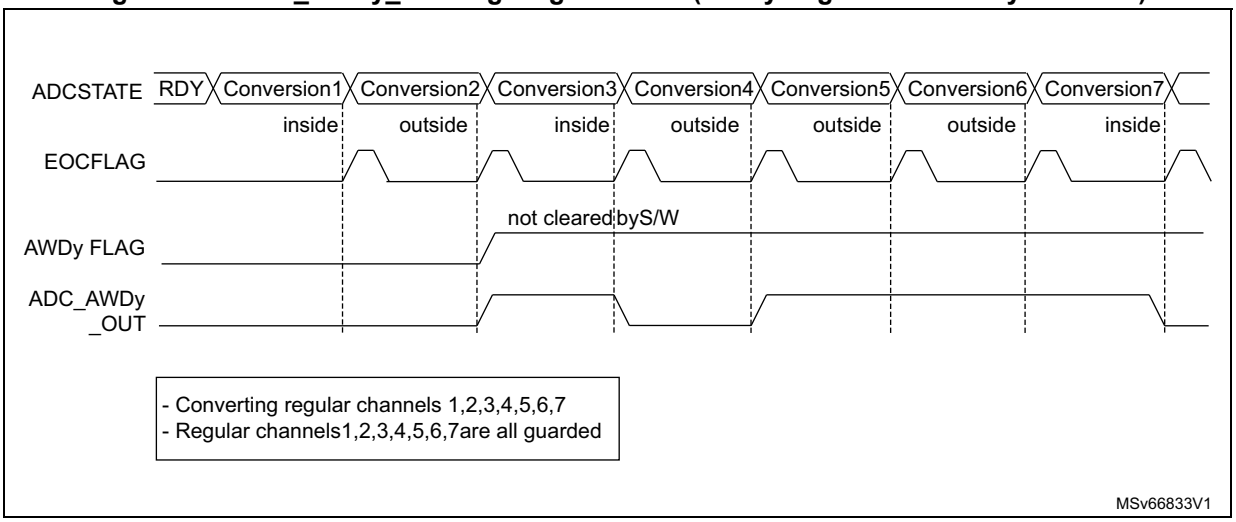


Figure 282. ADC_AWDy_OUT signal generation (on a single regular channel)

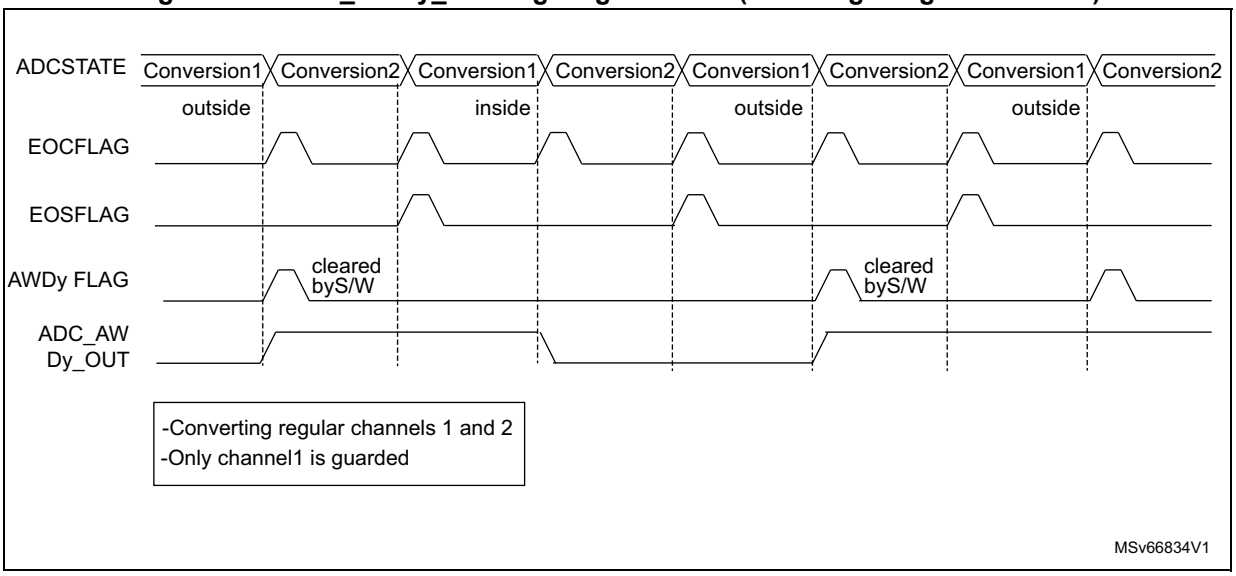
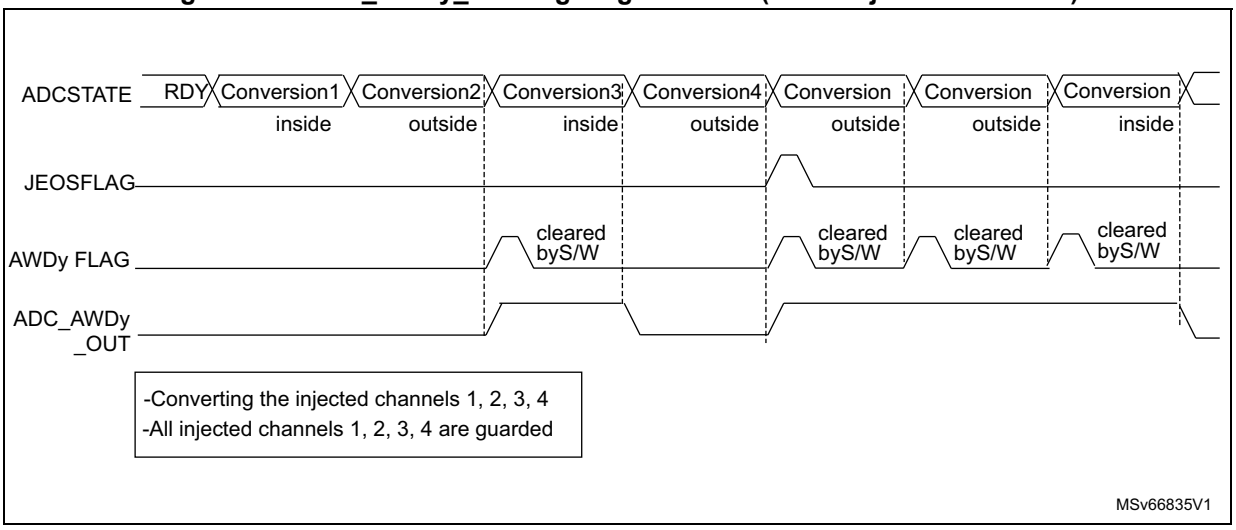


Figure 283. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control on the fly

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog filter for watchdog 1

With analog watchdog, a valid ADC conversion data range can be configured through the ADC_AWD1LTR and ADC_AWD1HTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWD1_OUT is generated as well as the AWD1 flag. Before this value is reached, AWD1 output is not activated.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

36.4.30 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22 bits.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_CFGR2 register, or by the JOVSS[3:0] bits in the ADC_CFGR3 register for parallel injected oversampling mode. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register.

The summation unit can yield a result up to 22 bits, which can be left or right shifted at the end. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

Figure 284 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 284. 12-bit result oversampling with 10-bit right shift and rounding

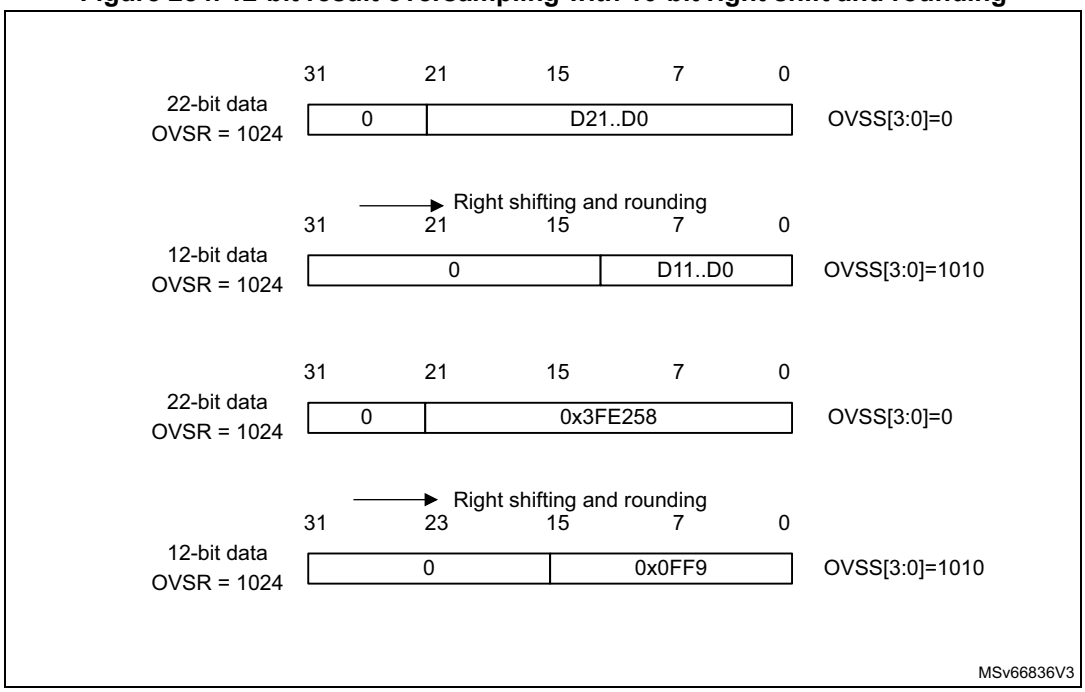


Table 272 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 316. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase for regular conversion.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the data latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and shift.
The EOC flag is set two additional clock cycles after the conversion is complete.

ADC operating modes supported in oversampling mode

In oversampling mode, most of the ADC operating modes are maintained:

- Single, discontinuous, or continuous conversion modes
- ADC conversions started either by software or triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits of the ADC_CFGR1 register) are accumulated, truncated, rounded, and shifted in the same way as 12-bit conversions are.
- Autoinjected mode (JAUTO) if JOVSPAR = 0 and oversampling ratio is the same for both regular and injected channels.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

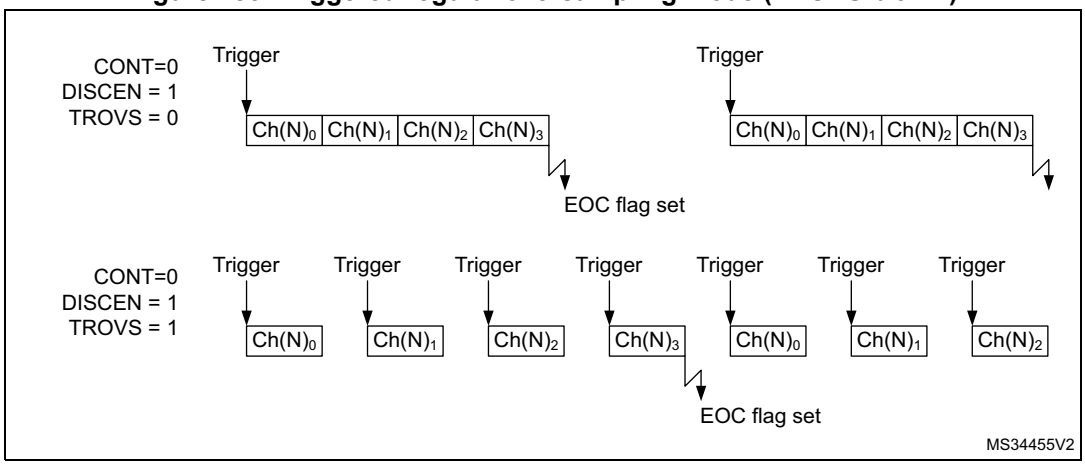
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 285 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set and OVSR ≠ 0, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither software trigger nor JAUTO is supported.

Figure 285. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

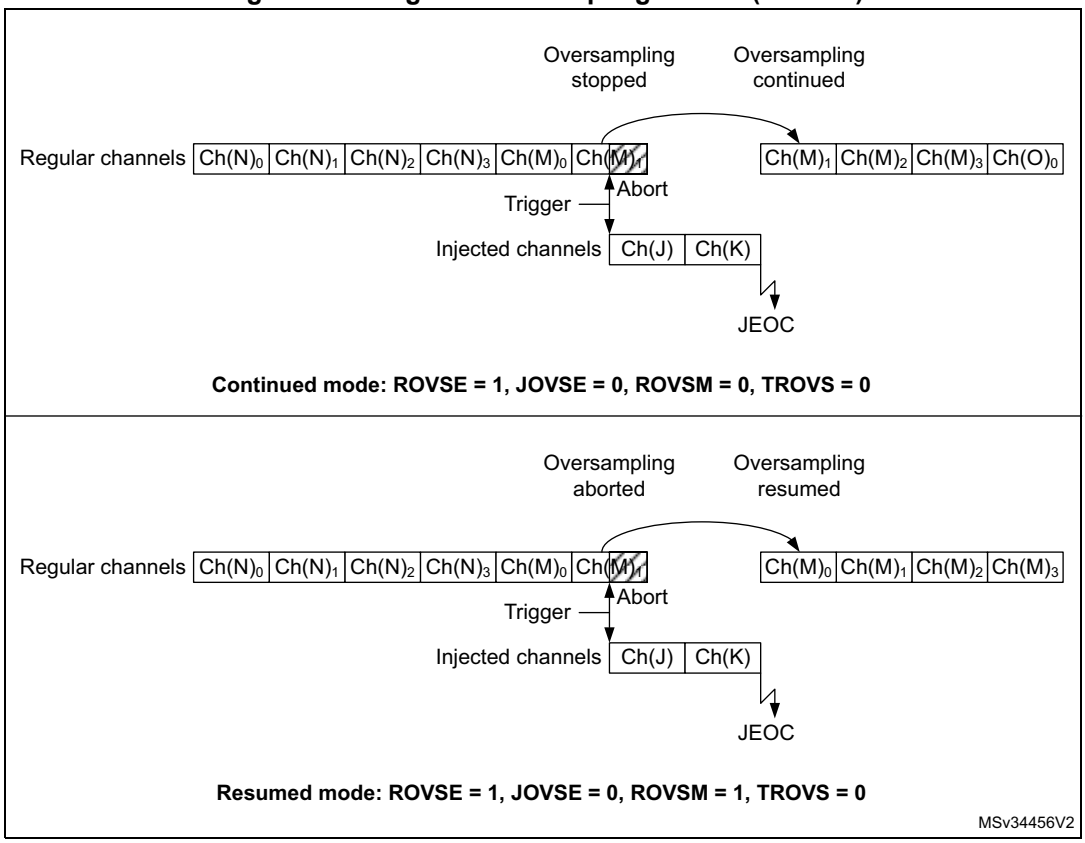
- In continuous mode (ROVSM = 0), the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that

oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);

- In resumed mode (ROVSM = 1), the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequence is blocked.

Figure 286 gives examples for a 4x oversampling ratio.

Figure 286. Regular oversampling modes (4x ratio)



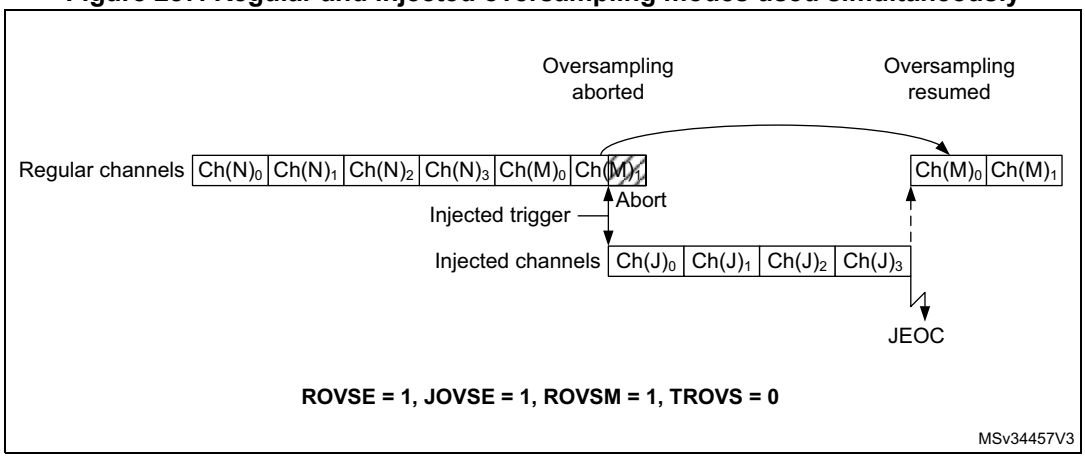
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 287*. This limitation no longer applies in parallel injected oversampling mode.

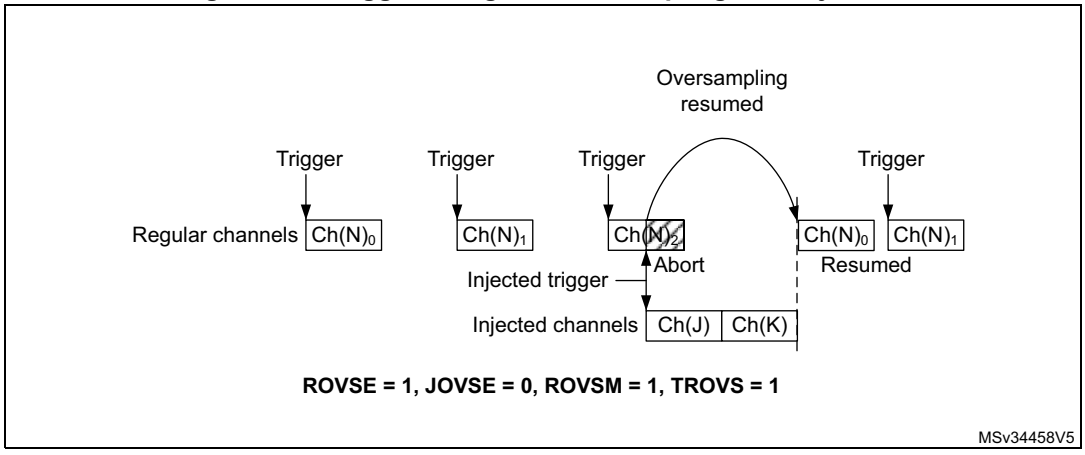
Figure 287. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected oversampling mode must be disabled and the ROVSM bit set (the resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 288*.

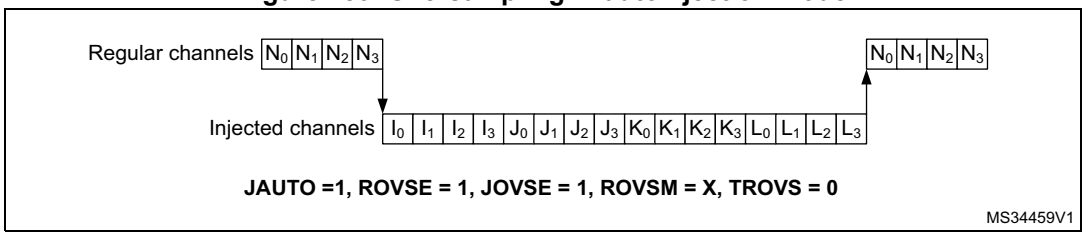
Figure 288. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 289* shows how the conversions are sequenced.

Figure 289. Oversampling in autoinjection mode



Combined modes summary

Table 317 summarizes all combinations, including the modes that are not supported.

Table 317. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

Table 318. DELAY bits versus ADC resolution(1)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$

Table 318. DELAY bits versus ADC resolution(1) (continued)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1100	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
others	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

1. Preliminary information, subject to change.

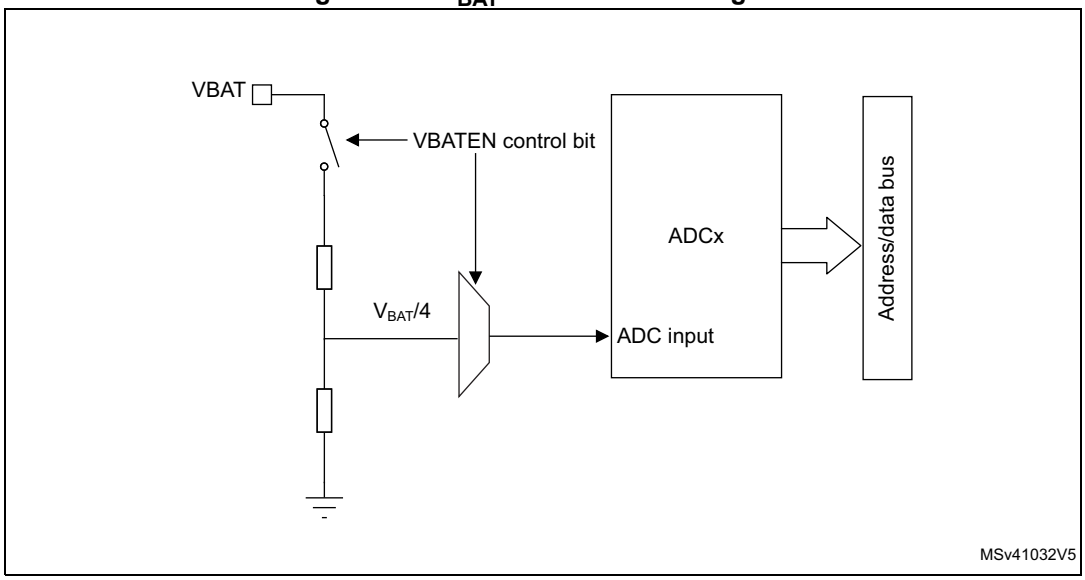
36.4.31 V_{BAT} supply monitoring

The VBATEN bit in the ADCC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage can be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 36.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 290 shows the block diagram of the V_{BAT} sensing feature.

Figure 290. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for V_{BAT}/4.

36.4.32 Monitoring the internal voltage reference

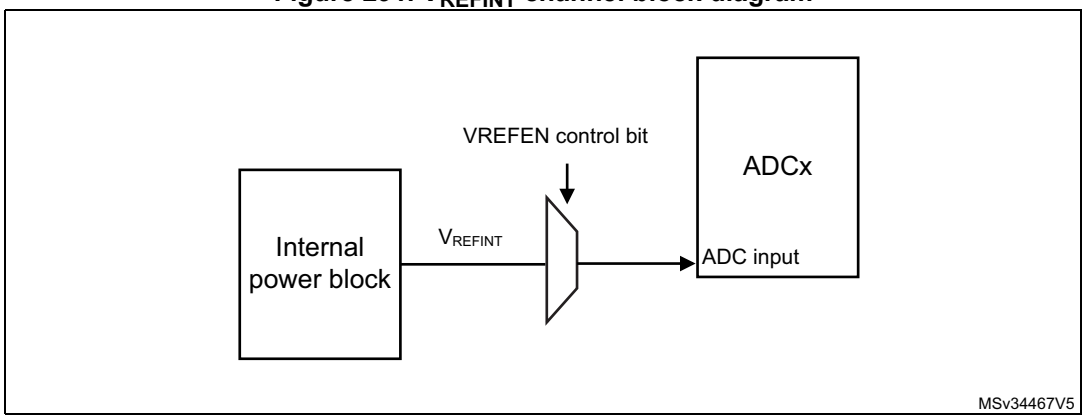
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 36.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 291 shows the block diagram of the V_{REFINT} sensing feature.

Figure 291. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value.
- ADC_DATA is the value measured by the ADC on channel x (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} - 1 = 4095$, or $2^8 - 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

36.4.33 Monitoring the supply voltage

The ADCs are connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

36.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2 and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 319. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			

36.6 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

36.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it or when ADEN = 0.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the corresponding ADC_JDRy register.

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it or when ADEN = 0.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it or when ADEN = 0.

0: Regular conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase. It is cleared by software writing 1 or when ADEN = 0.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it or when ADEN = 0.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

36.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

36.6.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to enter ADC calibration mode.

0: Normal mode

1: Calibration mode

Note: The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1, ADCAL = 1, ADSTART = 0 and JADSTART = 0 (ADC enabled, Calibration mode enabled and no conversion ongoing).

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential input mode for the calibration.

0: Single-ended input calibration mode when ADCAL = 1

1: Differential input calibration mode when ADCAL = 1

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bits 28:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (JEXTSEL = 0x0 and JDISEN = 1), this bit is cleared by hardware immediately after injected conversion starts.
- In all cases, after the execution of the JADSTP command, this bit and JADSTP are cleared by hardware simultaneously.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1, ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1), this bit is cleared by hardware immediately after regular conversion starts.
- In all cases, after the execution of the ADSTP command, this bit and ADSTP are cleared by hardware simultaneously.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

36.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 Reserved, must be kept at reset value.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0

00001: adc_ext_trg1

00010: adc_ext_trg2

00011: adc_ext_trg3

00100: adc_ext_trg4

00101: adc_ext_trg5

00110: adc_ext_trg6

00111: adc_ext_trg7

...

11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only

01: DMA one-shot mode selected

10: MDF mode selected

11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

36.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw					rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the BULB bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode
1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: 1-bit shift
0010: 2-bit shift
0011: 3-bit shift
0100: 4-bit shift
0101: 5-bit shift
0110: 6-bit shift
0111: 7-bit shift
1000: 8-bit shift
1001: 9-bit shift
1010: 10-bit shift
Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **ROVSE**: Regular oversampling enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled
1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

36.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

36.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

36.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **PCSEL[19:0]**: Channel i ($V_{INP[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{INP[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{INP[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Configuring the PCSEL bit is not necessary for the internal channels (such as VREFINT).

36.6.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **LEN[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

36.6.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

36.6.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

36.6.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

36.6.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular data converted
These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 36.4.26: Data management*.

36.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JLEN[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

36.6.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET_CH[4:0]					SSAT	USAT	POS OFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:0 Reserved, must be kept at reset value.

36.6.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:16]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

36.6.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

36.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 36.4.26: Data management*.

36.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.
AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2
AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2
When AWDCH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

36.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[19:0] = 000..0, the analog Watchdog 3 is disabled.

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

36.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1.

Refer to *Section 36.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*.

36.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD1 flag or an interrupt

001: two consecutive detections generate an AWD1 flag or an interrupt

...

111: Eight consecutive detections generate an AWD1 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 1. Refer to *Analog windows watchdog* section.

36.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

36.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold
These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

36.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold
These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

36.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

36.6.27 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential or pseudodifferential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential or pseudodifferential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

36.6.28 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALADDOS	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[8:0]								
rw							rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CALADDOS**: Calibration additional offset

This bit is set by software to add a positive offset at the input stage of ADC for calibration purpose.

0: Calibration additional positive offset disabled

1: Calibration additional positive offset enabled

Note: The software is allowed to write these bits only when ADSTART = 0, and JADSTART = 0 (ADC is enabled, calibration mode is selected, and no conversion is ongoing).

Bits 30:25 Reserved, must be kept at reset value.

Bits 24:16 **CALFACT_D[8:0]**: Calibration factors in differential mode

These bits are written by hardware or by software.

Once a differential input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new differential conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **CALFACT_S[8:0]**: Calibration factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new single-ended conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

This bit can only be cleared by a power-on reset.

36.6.29 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP3	OP2	OP1	OP0
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OP3:** Option bit 3

This bit is set and cleared by software to enable/disable the V_{DDCPU} channel.

0: V_{DDCPU} channel disabled

1: V_{DDCPU} channel enabled

Note: This bit is only available in ADC2

Bit 2 **OP2:** Option bit 2

This bit is set and cleared by software to enable/disable the V_{DDCORE} channel.

0: V_{DDCORE} channel disabled

1: V_{DDCORE} channel enabled

Note: This bit is only available in ADC2.

Bit 1 **OP1:** Option bit 1

This bit is set and cleared by software to choose the ADC bandgap source

0: ADC internal bandgap disabled

1: ADC internal bandgap enabled

Note: The software is allowed to write this bit only when ADC is disabled.

Bit 0 **OP0:** Option bit 0

This bit is set and cleared by software to enable the internal reference buffer.

0: ADC internal reference voltage buffer disabled

1: ADC internal reference voltage buffer enabled

Note: The software is allowed to write this bit only when ADC is disabled.

36.7 ADC common registers

These registers define the control and status registers common to both ADCs.

36.7.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

36.7.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	Res.	VREFEN	PRESC[3]	PRESC[2]	PRESC[1]	PRESC[0]	Res.	Res.
							rw		rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: V_{BAT} is not available on all ADC instances. Refer to Section 36.4.4: ADC connectivity for details.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: V_{REFINT} is not available on all ADC instances. Refer to Section 36.4.4: ADC connectivity for details.

Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Other: reserved, must not be used

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 17:0 Reserved, must be kept at reset value.

36.7.3 ADC hardware configuration register (ADCC_HWCFGR1)

Address offset: 0x0F0

Reset value: 0x0000 0241 (ADCC1)
Reset value: 0x0000 0241 (ADCC2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ANAIP[3:0]				OPBITS[3:0]				ADCNUM[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **ANAIP[3:0]**: Analog IP compatibility
0001: ADC12P18
0010: ADC12FF16/N12
0011: ADC12T40
0100: ADC12S40
0101: ADC12M40/E40
0110: ADC12M40/E40_2msps

Bits 7:4 **OPBITS[3:0]**: Number of option bits
0000: No option register implemented
0001: 1 option bit implemented in the ADC option register (ADC_OR) at address offset 0x0D0
0010: 2 option bits implemented in the ADC option register (ADC_OR) at address offset 0x0D0
0100: 4 option bits implemented in the ADC option register (ADC_OR) at address offset 0xD0

Bits 3:0 **ADCNUM[3:0]**: Number of ADCs implemented
0001: One ADC instance implemented
0010: Two ADC instances implemented
0011: Three ADCs instances implemented

36.7.4 ADC version register (ADCC_VERR)

Address offset: 0x0F4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits returns the ADC IP major revision
0001: Major revision = 1.X
0010: Major revision = 2.X
0011: Major revision = 3.X

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits returns the ADC IP minor revision
0000: Minor revision = X.0
0001: Minor revision = X.1

36.7.5 ADC identification register (ADCC_IPIDR)

Address offset: 0x0F8

Reset value: 0x0011 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Peripheral identifier
These bits returns the ADC identifier.
ID[31:0] = 0x0011 0008: c7amba_aditf6

36.7.6 ADC size identification register (ADCC_SIDR)

Address offset: 0x0FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size Identification
SID[31:8]: fixed code that characterizes the ADCx_SIDR register. This field is always read at 0xA3C5DD.
SID[7:0]: read-only numeric field that returns the address offset (in Kbytes) of the identification registers from the IP base address:
0x01: 1 Kbytes address offset
0x02: 2 Kbytes address offset
0x04: 4 Kbytes address offset
0x08: 8 Kbytes address offset

36.8 ADC register map

ADC1 and ADC2 are XXX.

Table 320. ADC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	ADCALDIF	DEEPPWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1																								0	0	0	0	0	0
0x00C	ADC_CFGR1	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	Reset value	0x0000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0					0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved																																
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	Res.	SQ5[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	Res.	SQ10[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]			
	Reset value																							0	0	0	0	0		0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 320. ADC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044-0x048	Reserved																																
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	ADC_OFCFGR1	OFFSET1_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2	OFFSET2_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3	OFFSET3_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4	OFFSET4_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074-0x07C	Reserved																																
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 320. ADC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090-0x09C	Reserved																																
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADC_CALFACT	CALADDOS	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[8:0]								
	Reset value	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved																																
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP3	OP2	OP1	OP0
	Reset value																													0	0	0	0
0x0D4-0x0FC	Reserved																																

Table 321. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	Res.	VREFEN	PRESC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0	0	0	0	0																		
x00C-0x0EC	Reserved																																
0x0F0	ADCC_ HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANAIP[3:0]				OPBITS[7:0]								ADCNUM[3:0]			
	Reset value	0x0000 0241 (ADCC1) 0x0000 0241 (ADCC2)																															
0x0F4	ADCC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0011																															
0x0F8	ADCC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0x0FC	ADCC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

37 Digital temperature sensor (DTS)

37.1 Introduction

The DTS is a high-precision low-power junction temperature sensor (TS). It is composed of a configurable controller plus a temperature sensor, TS0. The controller features a generic interface that enables the DTS to be accessed in read and write modes, through the APB bus.

37.2 DTS main features

The temperature sensor main features are the following:

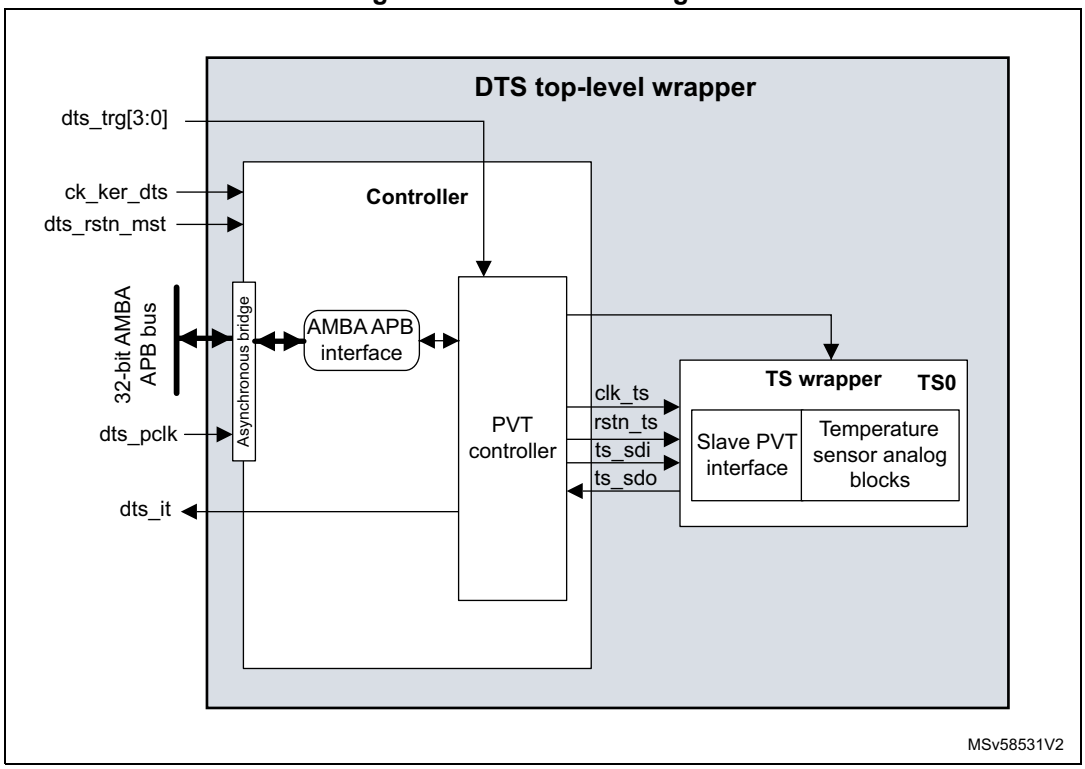
- For each sensor:
  - Two programmable (rise or fall) hardware alarms incorporating hysteresis
  - Status registers recording the minimum and maximum data values received
- A power-up timer with IRQ to support manual operation

37.3 DTS functional description

37.3.1 DTS block diagram

Figure 292 shows the DTS block diagram.

Figure 292. DTS block diagram



The DTS incorporates an APB interface configured to control the temperature sensor, TS0, through a process, voltage and temperature (PVT) controller.

37.3.2 DTS pins and internal signals

Table 322. DTS internal input/output signals

Internal signal name	Signal type	Description
dts_pclk	Input	APB bus clock
ck_ker_dts	Input	DTS main clock
dts_rstn_mst	Input	Master reset (asynchronous, active low)
AMBA APB bus	Input/output	APB Interface
dts_it	Output	IRQ interrupt from the DTS peripheral
dts_trg[3:0]	Inputs	Trigger interface
Temperature sensor (TS) interface		
clk_ts	Input	Temperature sensor SDA clock
rstn_ts	Input	Temperature sensor SDA reset (asynchronous, active low)
ts_sdi	Input	Temperature sensor SDA slave data Input
ts_sdo	Output	Temperature sensor SDA slave data output

37.3.3 DTS reset and clocks

DTS clocks

The DTS is clocked by ck_ker_dts.

The PVT controller incorporates a clock synthesizer in the TS block. The synthesizers generate the clock and timing strobes for the slave serial data adapters (SDA) and the clock for the embedded TS block (clk_ts).

The synthesizer CLK_SYNTH_HI[7:0]/CLK_SYNTH_LO[7:0] counters allow the integer division of the ck_ker_dts clock to generate the TS clock (clk_ts) at frequency ratios from 1/2 to 1/512 (see *DTS TSC clock synthesizer register (DTS_TSCCLKSYNTHR)*). Separate counters are used to generate timing control strobes that are aligned with the clk_ts clock. These timing strobes are used by the master serial adapters to control the sample point of the serial data inputs and the hold time of the serial data outputs.

The system clock ck_ker_dts frequency is determined by the RCC. CLK_SYNTH_HI[7:0] and CLK_SYNTH_LO[7:0] must be programmed so that the ratio CLK_SYNTH_HI[7:0]/CLK_SYNTH_LO[7:0] enables to obtain a clk_ts serial clock that falls within the TS operating frequency range (from 4 to 8 MHz).

The TS frequency is defined by:

$$\text{clk_ts frequency} = 1 / ((\text{CLK_SYNTH_HI[7:0]} + 1) + (\text{CLK_SYNTH_HI[7:0]} + 1)) \times \text{ck_ker_dts frequency}$$

For example:

For ck_ker_dts = 64.0 MHz and a TS operating frequency ranging between 4 and 8 MHz, setting the ck_ker_dts to clk_ts divider factor to 8:1 (CLK_SYNTH_HI[7:0] and CLK_SYNTH_LO[7:0] both programmed to 0x03), returns an TS operating frequency of 64.0 MHz / 8 = 8 MHz, which falls within the acceptable operating frequency range of 4 to 8 MHz. This enables to obtain an overall divider factor of 8 and a ck_ker_dts serial clock duty cycle of 50/50.

For ck_ker_dts = 8 MHz, the configuration must be:

CLK_SYNTH_LO[7:0] = 0x0

CLK_SYNTH_HI[7:0] = 0x0

CLK_SYNTH_HOLD[3:0]= 0x1

clk_ts = 4 MHz

clk_ts duty cycle = 50 %

The CLK_SYNTH_HOLD[3:0] value must be programmed to accommodate the worst-case serial data path propagation delays without exceeding the synthesized clock period, this means that:

$$\text{CLK_SYNTH_HOLD[3:0]} \leq \text{CLK_SYNTH_HI[7:0]} + \text{CLK_SYNTH_LO[7:0]} + 1$$

DTS reset

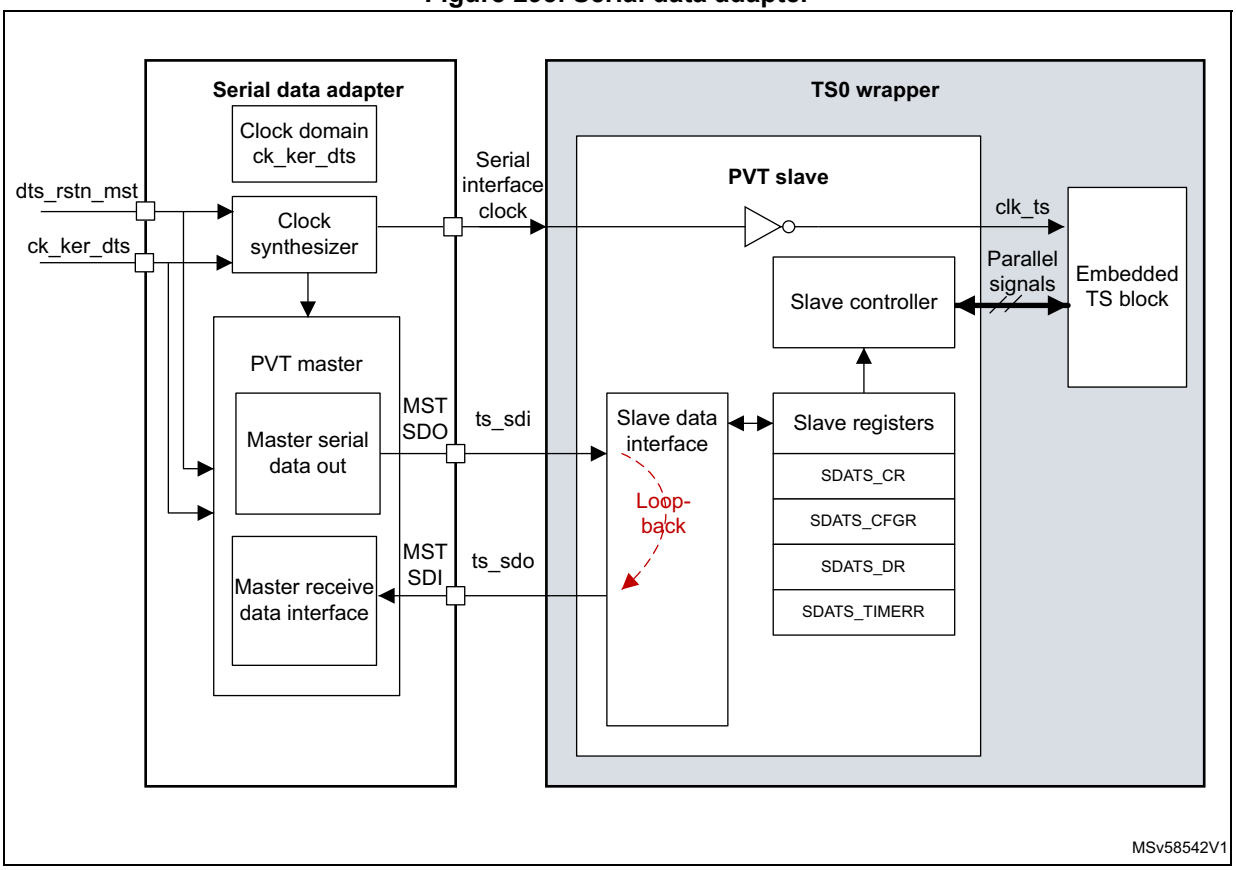
The DTS is reset using the asynchronous active-low dts_rstn_mst input.

When the master dts_rstn_mst is asserted, the entire DTS is reset, all APB internal registers and the PVT controller. It also powers down and resets the embedded TS blocks.

When de-asserted, the APB interface responds to requests after a three clock-cycle delay.

37.3.4 DTS serial data adapter (SDA)

Figure 293. Serial data adapter



The serial data adapter (SDA) allows the TS blocks to be configured and controlled via serial data interfaces (SDIF). The purpose of the SDA is to minimize the number of signals that need to be routed from the PVT controller to each TS block.

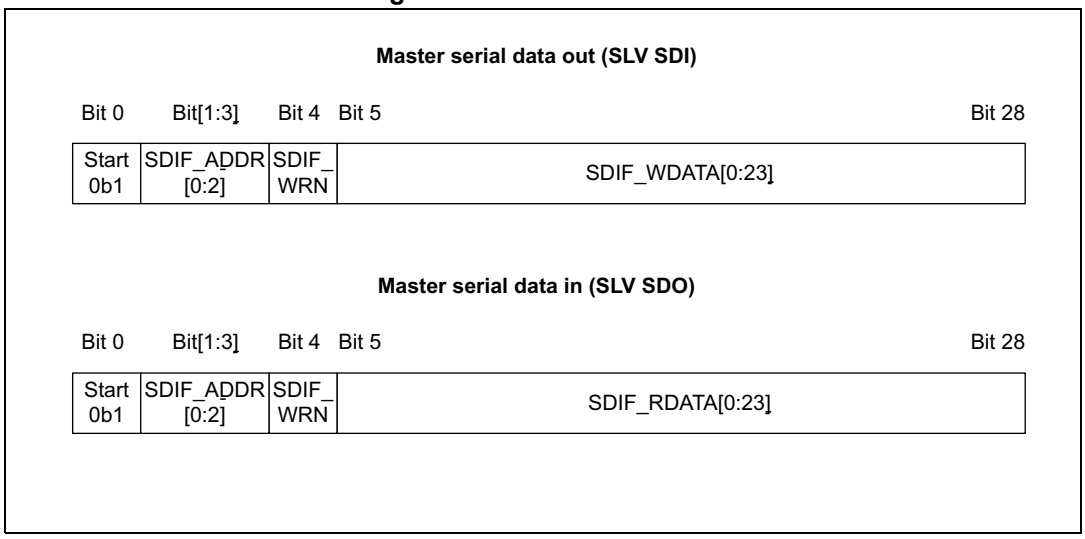
A master usually transmits data to all its active PVT slave devices. However, it is possible to inhibit the programming of one or more slaves to allow different configuration options to be applied. To achieve power saving, it is also possible to disable completely a TS block by disabling its clock and forcing reset.

The serial data interface also provides loop-back functionality (shown in red in *Figure 293*) to support system integration testing without having to power up the embedded TS block.

SDA data format

The serial data streams consist of an active-high start bit, 3-bit register address (SDIF_ADDR[2:0]), register write/not read enable (SDIF_WRN), and 24-bit data field (SDIF_WDATA[23:0]). The input and output serial data stream formats are shown in *Figure 294*. It must be noted that the serial data streams are little endian, meaning that the individual data fields are shifted LSB first.

Figure 294. SDA data format



SDA read

When reading data from the SDA slave registers (SDIF_WRN = 0), the SLV SDI bit stream is shifted to SLV SDO (delayed by one clk_ts cycle) and the register read data value is output on SDIF_RDATA[23:0]. The read data value is then made available in the PVT controller DTS_TS0SDIFRDATAR register.

SDA write

When writing data to the SDA slave registers (SDIF_WRN = 1), the SLV SDI bit stream is automatically shifted to SLV SDO and SDIF_RDATA[23:0] = SDIF_WDATA[23:0]. The write data loop-back value is then made available in the corresponding PVT controller DTS_TS0SDIFRDATAR register.

A special loop-back mode is available by writing to the (read-only) SDATS_DR register when SDIF_WRN = 1 and SDIF_ADDR[2:0] = 0x3. In this mode, the loop-back data value is also written to the PVT controller TS data registers (DTS_TS0SDIFRDATAR). When writing loop-back data to the TS data registers, the SDA TS_DATA[23:0] field **must** be set to 0x0.

The DTS_TS0SDIFRDATAR loop-back functionality can be used to verify the integrity of the serial data transfer between the PVT controller and remote PVT slave devices.

The special DTS_TS0SDIFRDATAR loop-back functionality is provided to test the PVT controller operation without having to power up the embedded TS block, for example when a fault or alarm occurs.

The loop-back functionality is disabled whenever the PVT slave is configured to automatic data recovery when the TS_AUTO bit of PVT SDATS_CR register is asserted.

SDA automatic data recovery

The automatic data recovery is enabled through TS_AUTO bit of SDATS_CR register.

In automatic data recovery mode, the SDIF_LOCK flag is set in the DTS_TSCSDIF_SR register, the SDA interface becomes write-only and all SDA read requests are ignored. In this mode, the SLV automatically streams TS_DATA[23:0] to the PVT controller each time new data samples become available, while setting SDIF_ADDR[2:0] to 0x3 and SDIF_WRN to 1. The automatically recovered data are made available in the PVT controller DTS_TS0SDIFRDATAR and DTS_TS0SDIFRDATAR register.

SDA SDIF programming

The SDA slave is accessed via the TS SDIF control register DTS_TSCSDIF_CFGR, by setting the SDIF_WDATA[23:0], the SDIF_ADDR[2:0] and the SDIF_WRN, and asserting the SDIF_PROG bit. This SDA read or write request is normally set up using a single PVT register write access.

Once a programming request has been issued, the SDIF_BUSY flag in the DTS_TSCSDIF_SR register is set, and any additional read or write programming requests is ignored until the current access completes.

SDA controller operation

The SDA slave controller features two operating modes: manual (default) and automatic. In both modes, the embedded TS block is configured through the SDATS_CFGR register.

In manual operating mode, when the TS_AUTO bit of SDATS_CR register is de-asserted, the embedded TS block is controlled directly through the SDATS_CR register fields TS_PD, TS_RSTN, TS_RUN_ONCE, TS_RUN_CONT and TS_CLOAD. In manual mode, the user is responsible for ensuring that the PD, RSTN, CLOAD and RUN signals are applied in the correct order and at appropriate times. Sample data from the TS block are captured in the SDA slave SDATS_DR register at the end of the conversion. The user must either poll the SDATS_DR register or leave sufficient time for the conversion to complete before reading the SDATS_DR register in order to recover the sample data from the slave SDA.

In automatic mode, when the TS_AUTO bit is asserted in SDATS_CR register, the embedded TS block is controlled by the slave SDA. When either TS_RUN_ONCE or TS_RUN_CONT bit of the TS_SDATS_CR register is asserted, the slave SDA sequences the PD, RSTN, CLOAD, and RUN signals that drive the TS block to perform one or more conversions. In this mode sample data from the TS block are captured in the SDA slave SDATS_DR register at the end of the conversion before being automatically streamed to the PVT controller.

In trigger mode, when a trigger is asserted at DTS top level, this value is compared to the value programmed in bits 18 to 16 (TRIG[2:0]) of the SDATS_CFGR register. If it is equal, the TS_RUN_ONCE bit is automatically asserted. Sample data from each TS block are captured in the SDA slave SDATS_DR register at the end of the conversion.

SDA slave registers

The serial data adapter registers are accessible through the SDIF interface of the PVT controller.

Table 323. SDA slave registers

SDIF_ADDR[2:0]	Reset value	Register	Description
0x0	0x000001	SDATS_CR	TS control register (see *Table 324*)
0x1	0x000000	SDATS_CFGR	TS configuration register (see *Table 325*)
0x3	0x000000	SDATS_DR	TS data / SDA loop-back register (see *Table 326*)
0x4	-	Reserved	-
0x5	0x000000	SDATS_TIMERR	TS power-up Timer register (see *Table 324*)

Table 324. SDA TS control register (SDATS_CR)

Bitfield range	Bitfield description	Access type
Bits 23:10	Reserved, must be kept at reset value.	-
Bit 9	**TS_STOP:** TS force stop This self-clearing bit immediately returns the slave TS controller to the default TS control state and terminates any serial data activity. This is not a clean shut-down and it may invalidate any data operations currently in progress. 0: TS not stopped 1: TS stopped	rs
Bit 8	**TS_AUTO:** TS automatic mode This bit is used in conjunction with TS_RUN_CONT and TS_RUN_ONCE to automatically power up the embedded TS and run conversions. Asserting TS_RUN_CONT triggers continuous conversions, while asserting TS_RUN_ONCE triggers a single conversion. If TS_AUTO is de-asserted while conversions are running, the controller waits for any active conversions to complete and then returns to the default TS control state. 0: Automatic mode disabled 1: Automatic mode enabled	rw
Bits 7:5	Reserved, must be kept at reset value.	-
Bit 4	**TS_CLOAD:** TS configuration load This self-clearing bit provides the embedded TS cload input in the default TS control state (TS automatic mode disabled). 0: TS configuration not loaded 1: TS configuration loaded	rs
Bit 3	**TS_RUN_CONT:** TS run continuous The TS_RUN_CONT control bit is ORed with TS_RUN_ONCE in order to provide the embedded TS run input in the default TS control state (TS automatic mode disabled). This bit is active high.	rw

Table 324. SDA TS control register (SDATS_CR) (continued)

Bitfield range	Bitfield description	Access type
Bit 2	**TS_RUN_ONCE:** TS run once Provided TS_RUN_CONT is not asserted, this self-clearing bit asserts the embedded TS run input for one clock cycle when it is in the default TS control state (TS automatic mode disabled). This bit is active on the rising edge.	rs
Bit 1	**TS_RSTN:** TS reset This bit provides the embedded TS rstn input in the default TS control state (TS automatic mode must be disabled). 0: TS reset 1: TS not reset	rw
Bit 0	**TS_PD:** TS power-down This bit provides the embedded TS pd input when in the default TS control state (TS automatic mode disabled). 0: No TS power-down 1: TS power-down	rw

Table 325. SDA TS configuration register (SDATS_CFGR)

Bitfield range	Bitfield description	Access type
Bits 23:20	Reserved, must be kept at reset value.	-
Bit 19	**CLKDIS:** TS clock gating disable bit 0: Clock enabled 1: Clock disabled	rw
Bits 18:16	**TRIG[2:0]:** Trigger selection bits 000: no trigger source selected 001: dts_trg[0] selected 010: dts_trg[1] selected 011: dts_trg[2] selected 100: dts_trg[3] selected Others: Reserved, must not be used Refer to *Section 37.3.7: Temperature sensor configuration* for details.	rw
Bits 15:8	Reserved, must be kept at reset value.	-

Table 325. SDA TS configuration register (SDATS_CFGR)

Bitfield range	Bitfield description	Access type
Bits 7:0	**TS0_CFG[7:0]:** TS0 configuration TS0_CFG[7:5]: TS0 resolution 000:12 bits 001:10 bits 010: 8 bits Other: reserved, must not be used TS0_CFGR[4]: TS0 output mode 0: parallel interface 1: serial interface (default) TS0_CFGR[3:0]: TS0 communication enable bit 0000: no communication 0001: communication enabled. Others: Reserved, must not be used.	rw

Table 326. SDA TS data register (SDATS_DR)

Bitfield range	Bitfield description	Access type
Bits 23:19	Reserved, must be kept at reset value.	-
Bit 18	**TS_DONE:** TS sample done flag This bit provides a way to identify when a new data sample has been recovered while manually polling the SDATS_DR register. In manually polling, the TS_DONE flag is cleared when the SDATS_DR register is read. The sample done flag is asserted when new sample data becomes available. It is automatically cleared in TS automatic mode as the SDATS_DR value is transmitted to the PVT controller. 0: No new data sample 1: New data sample	r
Bit 17	**TS_FAULT:** TS sample fault When asserted, this bit indicates that the embedded TS has signaled a fault. The TS_FAULT flag is updated when new data are received. 0: No TS sample fault 1: TS sample fault	r
Bit 16	**TS_TYPE:** TS sample type The TS_TYPE flag is updated when new data is received. 0: Indicates valid data 1 Indicates either analog access, signature select or fault debug mode	r
Bits 15:0	**TS_DATA[15:0]:** TS sample data	r

Table 327. SDA TS timer register (SDATS_TIMERR)

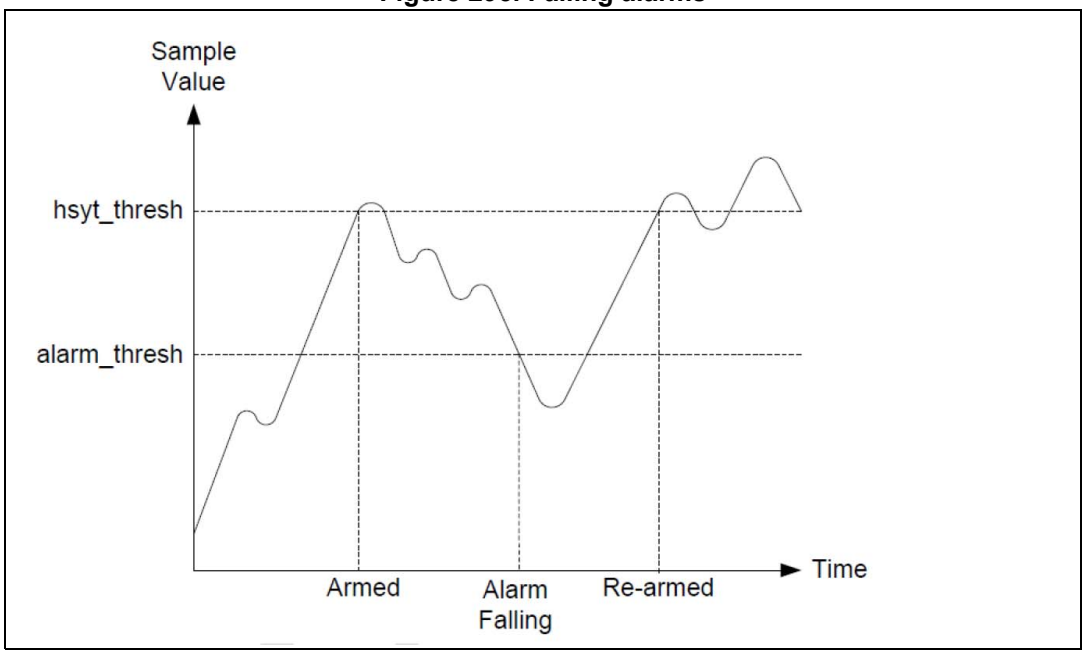
Bitfield range	Bitfield description	Access type
Bits 23:16	Reserved, must be kept at reset value.	-
Bits 15:0	**TS_PU_DLY[15:0]:** TS power-up delay The value is expressed in units of clk_ts cycles. Typical power-up delays are 256 cycles.	rw

37.3.5 Alarms

The DTS provides two alarms, alarm A and alarm B, for each TS block. The alarms feature programmable hysteresis and can be configured to detect either rising or falling sample values. They are disabled by setting the hysteresis threshold to a value that is equal to the alarm threshold (default condition). Once enabled, the alarms generate an interrupt if the value of the recovered data sample equals or exceeds the rising or falling alarm threshold. Once an alarm has been triggered, the hysteresis threshold provides a recovery window during which the alarm cannot trigger again.

Falling alarms

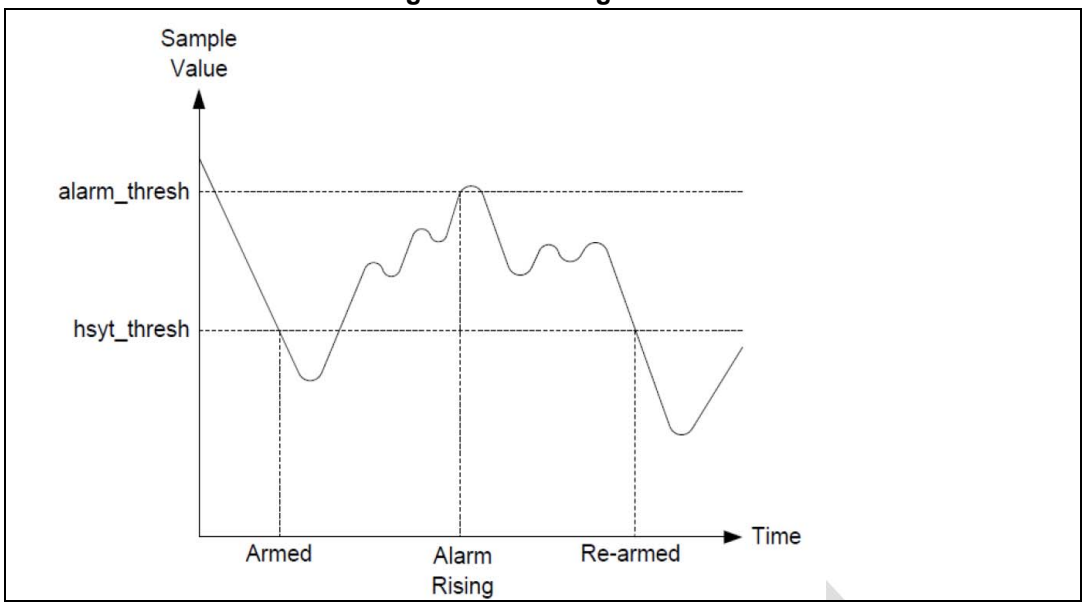
Figure 295. Falling alarms



Setting the hysteresis threshold greater than the alarm threshold creates a falling alarm. When configured as a falling alarm, the alarm is armed when the recovered sample value is greater than or equal to the hysteresis threshold. Once armed, a falling alarm triggers when the recovered sample value is less than or equal to the alarm threshold. An alarm is automatically disarmed once triggered and does not rearm until the recovered sample value is again greater than or equal to the hysteresis threshold. The alarm is also disarmed if an invalid sample value is received, that is if unexpected fault or type flags are detected.

Rising alarms

Figure 296. Rising alarms



Setting the hysteresis threshold less than the alarm threshold creates a rising alarm. When configured as a rising alarm, the alarm is armed when:

1. The associated clock synthesizer is first enabled (initial conversion).
2. The recovered sample value is less than or equal to the hysteresis threshold.
3. An invalid sample value is received, that is when an unexpected fault or type flags are detected.

Once armed, a rising alarm triggers when the recovered sample value is greater than or equal to the alarm threshold. The alarm is automatically disarmed once triggered and does not rearm until the recovered sample value is again less than or equal to the hysteresis threshold.

37.3.6 Main APB programming routines

This section gives several examples of routines to configure the PVT controller, program the SDA registers, and perform single or continuous temperature conversions.

Configure the PVT controller

1. STEP 1: Configure the PVT controller Interrupts:
   a) Write DTS_PVT_IER register:

      The timer is only intended to be used when using the TS block in bypass mode, so this interrupt must remain disabled .

   b) Write DTS_TS0_IER register:

      In the unlikely event of the TS block signaling a fault, the recovered data values are considered invalid and any active alarm is disarmed until the fault is removed. The fault IRQ is generated the first time a TS that an TS block signals a fault. If the fault is not cleared by the user, then no further interrupts are raised. It is recommended to enable always this interrupt.

2. STEP 2: Configure the PVT controller alarms:
   a) Write DTS_TS0ALARMA_CFGR register:

      Alarms are enabled when the ALARM_THRESH is not equal to the HYST_THRESH.

      When enabling alarms, make sure that the corresponding IRQ is also enabled.
   b) Write DTS_TS0ALARMB_CFGR register:

      Alarms are enabled when the ALARM_THRESH is not equal to the HYST_THRESH.

      When enabling alarms, make sure the corresponding IRQ is also enabled.
3. STEP 3: PVT sample control:
   a) Write DTS_TSCSMPL_CR register:

      When SMPL_DISCARD is set, TS data samples are temporarily discarded, but the sample counter continues to increment.

      When SMPL_CTR_HOLD is asserted, the sample counter is prevented from rolling over when the maximum sample count is reached.

      When SMPL_CTR_DISABLE is asserted, the sample counter is disabled.
4. STEP 4: PVT sample control:
   a) Write DTS_TS0HILORESETR for each TS instance.

Configure the SDA registers

1. STEP 1: Program the SDA registers:
   a) Read DTS_TSCSDIF_SR register:

      The SDIF_LOCK flag must be low, indicating that SDIF automatic mode has not been enabled.

      Before proceeding further, make sure that the serial data interface is inactive by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register to fetch the remote slave SDA register value:

      SDIF_PROG = 0x1

      SDIF_WRN = 0x1

      SDIF_ADDR[2:0]: this bitfield selects the desired SDA register.

      SDIF_WDATA[23:0]: this bitfield provides the SDA write data value.

      Program the SDA register bits in the following order:

      SDATS_TIMERR register

      SDATS_CFGR register

      SDATS_CR register (see notes below).

Note: Setting the TS_AUTO bit and either TS_RUN_ONCE or TS_RUN_CONT bit in the SDATS_CR register initiates a conversion and sets the SDIF_LOCK flag. Once the SDIF_LOCK flag is set, the SDA registers become write-only.

The SDIF_WDATA[23:0] value is also reflected in the DTS_TS0SDIFRDATAR *registers.*

2. STEP 2: Read the SDA register values:
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag must be low, indicating SDIF automatic mode has not been enabled.
      Before proceeding further, make sure the serial data interface is inactive by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register to fetch the remote slave SDA register value:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x0
      SDIF_ADDR[2:0]: this bitfield selects the desired SDA register.
      SDIF_WDATA[23:0]: this bitfield is not used and can be set to any value.

Note: Once the SDIF_LOCK flag is set, the SDA registers become write-only and any read requests is ignored.

The read data value (SDIF_RDATA[23:0]) is also reflected in the DTS_TS0SDIFRDATAR *register.*

   c) Read the SDA register value from the DTS_TS0SDIFRDATAR register.

Run once

1. STEP 1:
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag is low indicating that the SDIF automatic sample mode is not enabled.
      Verify that the serial data interface is not in use by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x1
      SDIF_ADDR[2:0] = 0x0 to select the SDATS_CR register
      SDIF_WDATA[23:0] = 0x000104 to assert TS_AUTO and the self-clearing TS_RUN_ONCE bits
2. STEP 2: Wait for a sample done IRQ
   a) Read the DTS_TS0SDIFDONER register:
      Verify that one or more SDATS_DR registers have been updated by checking that the SDIF_SMPL_DONE flags are set to 0x1.
   b) Read the DTS_TSCSMPLCNTR register:
      The sample counter value contained in DTS_TSCSMPLCNTR increments whenever TS data (TS_DATA[23:0]) are updated.
   c) Read the DTS_TS0SDIFRDATAR register:
      Read the TS sample data, type and fault values.
      If the fault bit is set, the sample data value is unreliable, and must be discarded.
      The SDIF_SMPL_DONE flag in the DTS_TS0SDIFDONER register is automatically cleared when the SDATS_DR register is read.

Continuous run

1. STEP 1
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag is low to indicate that the SDIF automatic sample mode is not enabled.
      Verify that the serial data interface is not in use by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x1
      SDIF_ADDR[2:0] = 0x0
      SDIF_WDATA[23:0] = 0x000108 (value to assert TS_AUTO and the run continuous TS_RUN_CONT bits)
2. STEP 2: Wait for a sample done IRQ
   a) Read the DTS_TS0SDIFDONER register:
      Verify that the SDATS_DR registers have been updated by checking that the sample done SDIF_SMPL_DONE flags are set to 0x1.
   b) Read DTS_TSCSMPLCNTR register:
      The sample counter increments whenever the DTS_TS0SDIFRDATAR value is updated.
   c) Read DTS_TS0SDIFRDATAR register:
      Read the TS sample data, type and fault values.
      If the fault bit is set, the sample data value is unreliable, and must be discarded.
      The SDIF_SMPL_DONE flag in the DTS_TS0SDIFDONER register is automatically cleared when the SDATS_DR register is read.

37.3.7 Temperature sensor configuration

The temperature sensor can be configured through the internal register *SDA TS configuration register (SDATS_CFGR)*.

Selecting the conversion mode

The sensor operates only in a single mode to read the temperature value. This mode is selected by setting SDATS_CFGR[3:0] bits to 0b0001.

The temperature value is obtained using the formula provided in *Calculating the temperature*. It does not require any calibration.

Configuring the output options

The conversion time can be configures through SDATS_CFGR[7:5] bits. It can be shortened to reduce the counter output value with the drawback of degrading the temperature measurement accuracy (see *Table 328*).

The output data are available in the TS_DATA[15:0] bitfield of the SDATS_DR register.

Data can be output in parallel or serial mode depending on the setting of SDATS_CFGR[4] bit:

- 0: parallel output

  All the data bits are sent simultaneously through TS_DATA[11:0].
- 1: serial output (default mode)

  The data are sent serially on bit 11 of the TS_DATA bitfield.

  Serial data are always 12-bit wide whatever the output options. Using a lower resolution (such as 10 bits) results in similar output values with trailing zeroes.

Table 328. Output resolution configuration

Resolution	SDATS_CFGR[7:5] bits	Accuracy degradation	Comments
12 bits	0b000	-	Temperature value available on TS_DATA[11:0]
10 bits	0b001	±0.4 °C	Temperature value available on TS_DATA[11:2]
8 bits	0b010	±1.5 °C	Temperature value available on TS_DATA[11:4]
Others not allowed	0b011	-	Not allowed
	0b1XX	-	Not allowed

Calculating the temperature

The digital output is converted to an uncalibrated temperature value by using the following formula:

$$\text{Temp (°C)} = G + H \times \text{Eqbs} + J \times F_{clk_ts}$$

Where:

Temp = Temperature in degrees Celsius

Eqbs = Nbs/Cal5 – 0.5, Nbs being the digital output, TS_DATA[11:0], latched after conversion

F_{clk_ts} = Temperature sensor SDA clock frequency (in MHz)

For G, H, J and Cal5 values, please refer to device datasheet"

Configuring the triggers

Triggers can be configured through bits 18 to 16 (TRIG[2:0]) of the SDATS_CFGR register:

TRIG[2:0] = 0b001: dts_trg[0] is selected.

TRIG[2:0] = 0b010: dts_trg[1] is selected.

TRIG[2:0] = 0b011: dts_trg[2] is selected.

TRIG[2:0] = 0b100: dts_trg[3] is selected.

Disabling the clock

The TS clock can be disabled by setting the CLKDIS of the SDATS_CFGR, when the TS is not used and it does not operate in automatic mode.

37.4 DTS interrupts

The DTS provides a single active high (level) hardware interrupt output from the controller. This is the logical OR of all the embedded TS interrupt sources and the common delay timer.

The enabling, masking, and status of the interrupts are readily available through the IRQ control registers. The interrupts from each TS instance are coalesced into a single interrupt, the status of which can be observed through the DTS_TS0_ISR register. The index number of a set bit indicates the source TS number with an interrupt condition.

Each TS block can generate an interrupt based on the following sources:

- The irq_smpl indicates that sample data have been updated for the corresponding TS instance.
- The irq_alarma indicates that the recovered sample value has exceeded the programmed alarm A threshold for the corresponding TS instance.
- The irq_alarmb indicates that the recovered sample value has exceeded the programmed alarm B threshold for the corresponding TS instance.
- The irq_fault indicates that the corresponding TS instance has signaled a fault.

When an interrupt event occurs, the rising edge is detected and the corresponding flag is set in the interrupt source register, DTS_TS0_ISR. The IRQ status bits are individually gated (logical AND) with their corresponding interrupt enabled in the DTS_TS0_IER register to generate interrupt request bits. These IRQ bits are then ORed together and registered by ck_ker_dts to generate the interrupt outputs for IRQ control registers.

Once set the interrupt source bits are sticky, they must be cleared by writing a 1 to the corresponding bit of the DTS_TS0_ICR register. This also clears the associated IRQ if it has been enabled.

Interrupt events are triggered when the following flags are asserted or the force bits in the DTS_TS0IRQTESTR register are set. The forcing allows interrupts to be manually generated for software test purposes.

- irq_smpl: asserted whenever a new TS data sample is recovered.
- irq_alarma: asserted whenever the alarm A thresholds have been reached.
- irq_alarmb: asserted whenever the alarm B thresholds have been reached.
- irq_fault: asserted whenever the TS block asserts its fault output.

Generally, the embedded temperature sensors run continuously relying on their alarms to signal out-of-bounds conditions. The user would usually disable the sample done flags for these TS types unless they wish to log every data sample.

Table 329. DTS interrupts

Interrupt event	Event flag	Enable control bit	Interrupt clear method
irq_smpl	IRQ_STATUS_DONE	IRQ_EN_DONE	IRQ_CLEAR_DONE
irq_alarma	IRQ_STATUS_ALARMA	IRQ_EN_ALARMA	IRQ_CLEAR_ALARMA

Table 329. DTS interrupts

Interrupt event	Event flag	Enable control bit	Interrupt clear method
irq_alarmb	IRQ_STATUS_ALARMB	IRQ_EN_ALARMB	IRQ_CLEAR_ALARMB
irq_fault	IRQ_STATUS_FAULT	IRQ_EN_FAULT	IRQ_CLEAR_FAULT

37.5 DTS registers

The APB interface supports 32-bit word aligned addresses (byte access is not supported).

The memory map contains a mixture of read-only, read-write, and write-only register fields. The write-only bits are self-clearing bits that are generally used to create strobes. Reserved register fields and any undefined register addresses are read-only and return zero when read.

Table 330 shows the DTS memory map overview.

Table 330. DTS register memory map

Address boundary	Register bank	Register block size	Address locations	Description
0x0000 - 0x003F	PVT common registers	0x40	16	PVT common registers
0x0040 - 0x007F	PVT interrupt registers	0x40	16	PVT interrupt registers
0x0080 - 0x00BF	TS common registers	0x40	16	TS common registers
0x00C0 - 0x00FF	TS 0 individual registers	0x40	16	TS0 dedicated registers (such as IRQ, alarms and samples)

37.5.1 DTS PVT common registers

The PVT common registers are used to configure the timer and read back the lock status.

DTS PVT register lock register (DTS_PVTREG_LOCKR)

The DTS can be locked by software using the DTS_PVTREG_LOCKR register. Any write to the register locks all DTS registers except for the PVT common registers.

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **LOCK[31:0]:** PVT software lock register
Any write to this register locks all DTS registers except for the PVT common registers. Once locked, the DTS registers are read-only. The software lock can be removed by writing 0x1ACC E551.

DTS PVT lock status register (DTS_PVTLOCK_SR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LOCK_STATUS	SW_LOCK_STATUS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HW_LOCK_STATUS:** Hardware lock input status
This bit indicates the status of the hardware lock input to the PVT controller.

Bit 0 **SW_LOCK_STATUS:** Software lock input status
This bit indicates the current state of the lock register.

DTS PVT timer control register (DTS_PVTTMR_CR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_RUN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR_DELAY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TMR_RUN:** Timer count enable bit
This bit enables the timer count function.

Bits 15:0 **TMR_DELAY[15:0]:** Timer delay
This bitfield can be used to configure the timer delay.

DTS PVT timer status register (DTS_PVTTMR_SR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_DONE	TMR_BUSY
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **TMR_DONE:** Counter delay expiration flag
This bit is set when the counter delay has expired.

Bit 0 **TMR_BUSY:** Counter busy flag
This bit is set while the counter is active.

37.5.2 PVT IRQ registers

The PVT IRQ registers are present in all build configurations of the PVT controller.

DTS PVT IRQ enable register (DTS_PVT_IER)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_IRQ_ENABLE	TMR_IRQ_ENABLE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **TS_IRQ_ENABLE:** TS IRQ source enable bit
Set this bit to enable the TS IRQ to the processor.

Bit 0 **TMR_IRQ_ENABLE:** Timer IRQ source enable bit
Set this bit to enable the timer IRQ to the processor.

DTS PVT IRQ timer mask register (DTS_PVTIRQTRMASKR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_MASK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TMR_IRQ_MASK:** Timer IRQ source mask bit
Set this bit to mask the timer IRQ to the processor.

DTS PVT IRQ TS mask register (DTS_TS_MR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0_IRQ_MASK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TS0_IRQ_MASK:** TS0 IRQ source mask bit
Set this bit to mask TS0 IRQ to the processor.

DTS PVT IRQ timer status register (DTS_PVTTR_SR)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_STATUS
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TMR_IRQ_STATUS:** Timer IRQ status bit after masking
This bit reflects the timer IRQ status after the mask has been applied.

DTS PVT IRQ TS status register (DTS_TS_ISR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0_IRQ_STATUS
															r

Bits 1 Reserved, must be kept at reset value.

Bit 0 **TS0_IRQ_STATUS:** TS0 IRQ status bit after masking
This bit reflects the timer IRQ status after the mask has been applied.

Note: The bit is set when the corresponding TS block has issued an IRQ interrupt to the processor.

DTS PVT IRQ timer raw status register (DTS_PVTTMRRAW_ISR)

Address offset: 0x070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_RAW_STATUS
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TMR_IRQ_RAW_STATUS:** TMR IRQ status bit before masking
This bit reflects the timer IRQ status before the mask is applied.

DTS PVT IRQ TS raw status register (DTS_TSRAW_ISR)

Address offset: 0x074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0_IRQ_RAW_STATUS
															r

Bits 1 Reserved, must be kept at reset value.

Bit 0 **TS0_IRQ_RAW_STATUS:** TS0 IRQ status bit before masking
This bit reflects the timer IRQ status before the mask is applied.

37.5.3 TS common registers

DTS TSC clock synthesizer register (DTS_TSCCLKSYNTHR)

Address offset: 0x080

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLK_SYTH_EN	Res.	Res.	Res.	Res.	CLK_SYNTH_HOLD[3:0]			
							rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK_SYNTH_HI[7:0]								CLK_SYNTH_LO[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CLK_SYTH_EN:** Synthesized clk_ts enable bit
0: Disabled
1: Enabled

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CLK_SYNTH_HOLD[3:0]:** SDA master-to-SDA slave output hold delay/SDA slave-to-SDA master input setup delay
The delay is counted from the rising edge of clk_ts. It is expressed in units of ck_ker_dts cycles. The default value is 0b0001.

Bits 15:8 **CLK_SYNTH_HI[7:0]:** Synthesized clk_ts high period
This is a zero-Indexed value expressed in units of ck_ker_dts cycles.

Bits 7:0 **CLK_SYNTH_LO[7:0]:** Synthesized clk_ts low period
This is a zero-Indexed value expressed in units of ck_ker_dts cycles.

DTS TSC SDIF interface disable register (DTS_TSCSDIFDISABLER)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	TS0_SDIF_DISABLE
															rw

Bits 1 Reserved, must be kept at reset value.

Bit 0 **TS0_SDIF_DISABLE:** TS0 serial data interface (SDIF) disable bit

When this bit is asserted, it completely disables the selected TS0 instance by forcing TS0 clock and reset low.

0: Enabled

1: Disabled

Note: Any disable action must be applied before SDIF programming begins.

DTS TSC SDIF status register (DTS_TSCSDIF_SR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_LOCK	SDIF_BUSY
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SDIF_LOCK:** SDIF locked flag

This bit is set when the slave automatic mode is requested, to indicate that the serial data interface (SDIF) is now a write-only interface (see *Section : SDA controller operation*).

In slave automatic mode, the data register (SDATS_DR) is automatically updated within the PVT controller whenever new TS data become available

0: Unlocked

1: Locked

Bit 0 **SDIF_BUSY:** SDIF busy flag

When asserted, this bit indicates either that the serial data interface (SDIF) is busy and any SDIF programming requests are ignored, or the clock synthesizer/disable SDIF state is being updated.

0: SDIF not busy/no clock synthesizer/disable SDIF state update ongoing

1: SDIF busy/clock synthesizer/disable SDIF state update ongoing

DTS TSC SDIF register (DTS_TSCSDIF_CR)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDIF_PROG	Res.	Res.	Res.	SDIF_WRN	SDIF_ADDR[2:0]			SDIF_WDATA[23:16]							
w				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDIF_WDATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SDIF_PROG:** Serial interface program request

This self-clearing bit is ignored if the SDIF_BUSY or the SDIF_LOCK flag is set, and the request is a read request (SDIF_WRN = 0).

0: No SDIF program request

1: SDIF program requested

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **SDIF_WRN:** Serial interface write/no read control bit

0: SDIF read

1: SDIF write

Bits 26:24 **SDIF_ADDR[2:0]:** Serial interface register address

This bitfield contains the address of the target SDIF register, SDIF_ADDR[2:0].

Bits 23:0 **SDIF_WDATA[23:0]:** Serial interface write data

This bitfield contains the write data SDIF_DATA[23:0].

DTS TSC SDIF halt register (DTS_TSCSDIFHALTR)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_STOP
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SDIF_STOP:** Serial data interface (SDIF) stop
This self-clearing bit immediately stops all SDIF programming activity in the unlikely event of it becoming locked. It must not be required during normal operation.
0: Not stopped
1: Stopped

DTS TSC SDIF control register (DTS_TSCSDIF_CFGR)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_INHIBIT[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SDIF_INHIBIT[1:0]:** Serial data interface (SDIF) programming inhibit
When asserted, this bit inhibits the serial programming of the selected TS instance(s). The inhibition must be applied before SDIF programming begins.
00: No Inhibition
01: TS0 inhibited
10: TS0 serial programming activated (SPA)
11: TS0 inhibited

DTS TSC sample control register (DTS_TSCSMPL_CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_DISCARD	SMPL_CTR_HOLD	SMPL_CTR_DISABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SMPL_DISCARD:** Sample discard bit

This feature can be used to temporarily discard received data samples. When this bit is set, data recovery is disabled but the external TS may continue running.

0: Data samples not discarded

1: Data samples discarded

Bit 1 **SMPL_CTR_HOLD:** Sample counter hold bit

When this bit is asserted, the sample counter is prevented from rolling over when the maximum sample count is reached.

0: Counter not on hold

1: Counter on hold

Bit 0 **SMPL_CTR_DISABLE:** Sample counter disable bit

0: Sample counter enabled

1: Sample counter disabled

DTS TSC sample clear register (DTS_TSCSDIFSMPLCLRR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_CNTER_CLEAR
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SMPL_CNTER_CLEAR:** Sample counter clear bit

The sample counter value is set to zero when this self-clearing bit is programmed to 1.

0: Counter value not cleared

1: Counter value cleared

DTS TSC sample count register (DTS_TSCSMPLCNTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPL_COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SMPL_COUNT[15:0]:** Sample counter

The 16-bit counter increments each time a new data sample is available. The counter is cleared by writing 1 to DTS_TSCSDIFSMPLCLRR. The sample counter overflow operation is determined by SMPL_CTR_HOLD.

37.5.4 TS individual registers

DTS TS0 IRQ enable register (DTS_TS0_IER)

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_EN_ALARMB	IRQ_EN_ALARMA	Res.	IRQ_EN_DONE	IRQ_EN_FAULT
											rw	rw		rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **IRQ_EN_ALARMB:** Alarm B IRQ enable bit

Bit 3 **IRQ_EN_ALARMA:** Alarm A IRQ enable bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_EN_DONE:** Sample done IRQ enable bit

Bit 0 **IRQ_EN_FAULT:** Fault IRQ enable bit

DTS TS0 IRQ status register (DTS_TS0_ISR)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_STATUS_ALARMB	IRQ_STATUS_ALARMA	Res.	IRQ_STATUS_DONE	IRQ_STATUS_FAULT
											r	r		r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **IRQ_STATUS_ALARMB:** Alarm B IRQ status bit

Bit 3 **IRQ_STATUS_ALARMA:** Alarm A IRQ status bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_STATUS_DONE:** Sample done IRQ status bit

Bit 0 **IRQ_STATUS_FAULT:** Fault IRQ status bit

DTS TS0 IRQ clear register (DTS_TS0_ICR)

Address offset: 0x0C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_CLEAR_ALARMB	IRQ_CLEAR_ALARMA	Res.	IRQ_CLEAR_DONE	IRQ_CLEAR_FAULT
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **IRQ_CLEAR_ALARMB:** Alarm B IRQ clear bit

Bit 3 **IRQ_CLEAR_ALARMA:** Alarm A IRQ clear bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_CLEAR_DONE:** Sample done IRQ clear bit

Bit 0 **IRQ_CLEAR_FAULT:** Fault IRQ clear bit

DTS TS0 IRQ test register (DTS_TS0IRQTESTR)

Address offset: 0x0CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_TEST_ALARMB	IRQ_TEST_ALARMA	Res.	IRQ_TEST_DONE	IRQ_TEST_FAULT
											rw	rw		rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **IRQ_TEST_ALARMB:** Alarm B IRQ test bit

Bit 3 **IRQ_TEST_ALARMA:** Alarm A IRQ test bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_TEST_DONE:** Sample done IRQ test bit

Bit 0 **IRQ_TEST_FAULT:** Fault IRQ test bit

DTS TS0 SDIF RDATA register (DTS_TS0SDIFRDATAR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_RDATA[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDIF_RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SDIF_RDATA[23:0]:** SDIF read data

DTS TS0 SDIF done register (DTS_TS0SDIFDONER)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_SMPL_DONE
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SDIF_SMPL_DONE:** Sample done flag

The sample done flag is asserted when a new data sample is available. The sample done flag is automatically cleared when the associated data register is read.

0: No new data sample available

1: New data sample available

DTS TS0 SDIF data register (DTS_TS0SDIFDATAR)

Address offset: 0x0D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_FAULT	SAMPLE_TYPE
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAMPLE_DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SAMPLE_FAULT:** Sample fault

This bit is set to indicate that the TS block has signaled a fault. The sample data must be discarded.

Bit 16 **SAMPLE_TYPE:** TS sample type

0: Indicates valid data

1: Indicates either analog access, signature select or fault debug mode

The TS_TYPE flag is updated when new data are received.

Bits 15:0 **SAMPLE_DATA[15:0]:** Sample data.

Reading this register clears the state of the SDIF_SMPL_DONE flag in DTS_TS0SDIFDONER register.

DTS TSn alarm A configuration register (DTS_TS0ALARMA_CFGR)

Address offset: 0x0E0 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALARMA_THRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HYSTA_THRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **ALARMA_THRESH[15:0]:** Alarm A threshold

Bits 15:0 **HYSTA_THRESH[15:0]:** Alarm A hysteresis threshold

DTS TSn alarm B configuration register (DTS_TS0ALARMB_CFGR)

Address offset: 0x0E4 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALARMB_THRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HYSTB_THRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **ALARMB_THRESH[15:0]:** Alarm B threshold

Bits 15:0 **HYSTB_THRESH[15:0]:** Alarm B hysteresis threshold

DTS TS0 high/low sample register (DTS_TS0HLSAMPLER)

Address offset: 0x0E8

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPL_HI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPL_LO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **SMPL_HI[15:0]:** Highest valid data sample value received

Bits 15:0 **SMPL_LO[15:0]:** Lowest valid data sample value received

DTS TS0 high/low reset register (DTS_TS0HILORESETR)

Address offset: 0x0EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_HI_CLR	SMPL_LO_SET
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SMPL_HI_CLR:** Sample high clear 0

All the bits of the stored sample high value are initialized to 0 when written.

Bit 0 **SMPL_LO_SET:** Sample Low Set

All the bits of the stored sample low value are initialized to 1 when written.

37.5.5 DTS register map

Table 331. PVT common register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000-0x00C	Reserved	Reserved																															
0x010	DTS_PVTREG_LOCKR	LOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DTS_PVTLOCK_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LOCK_STATUS	SW_LOCK_STATUS
	Reset value																															0	0
0x018 - 0x01C	Reserved	Reserved																															
0x020	DTS_PVTTMR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_RUN	TMR_DELAY[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DTS_PVTTMR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_DONE	TMR_BUSY
	Reset value																															0	0
0x030 - 0x03C	Reserved	Reserved																															

Table 332. PVT IRQ register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	DTS_PVT_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_IRQ_ENABLE	TMR_IRQ_ENABLE
	Reset value																															0	0
0x044 - 0x04C	Reserved	Reserved																															
0x050	DTS_PVTIRQTRMASKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_MASK
	Reset value																																0
0x054	DTS_TS_MR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0_IRQ_MASK
	Reset value																																0

Table 332. PVT IRQ register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058 - 0x05C	Reserved	Reserved																															
0x060	DTS_PVTTR_SR	Res.																															TMR_IRQ_STATUS
	Reset value																																0
0x064	DTS_TS_ISR	Res.																															TS0_IRQ_STATUS
	Reset value																																0
0x068 - 0x06C	Reserved	Reserved																															
0x070	DTS_PVTTMR RAW_ISR	Res.																															TMR_IRQ_RAW_STATUS
	Reset value																																0
0x074	DTS_TSRAW_ISR	Res.																															TS0_IRQ_RAW_STATUS
	Reset value																																0
0x078 - 0x07C	Reserved	Reserved																															

Table 333. DTS TSC clock synthesizer register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	DTS_TSCCLK SYNTHR	Res.							CLK_SYTH_EN	Res.				CLK_SYNTH _HOLD[3:0]				CLK_SYNTH_HI[7:0]								CLK_SYNTH_LO[7:0]								
	Reset value								0					0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 333. DTS TSC clock synthesizer register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	DTS_TSCSDIF DISABLER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0_SDIF_DISABLE
	Reset value																															0	0
0x088	DTS_TSCSDIF_ SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_LOCK	SDIF_BUSY
	Reset value																															0	0
0x08C	DTS_TSCSDIF_ CR	SDIF_PROG	Res.	Res.	Res.	SDIF_WRN	SDIF_ADDR [2:0]			SDIF_WDATA[23:0]																							
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	DTS_TSCSDIF HALTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_STOP
	Reset value																																0
0x094	DTS_TSCSDIF_ CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_INHIBIT [1:0]	
	Reset value																															0	0
0x098 - 0x09C	Reserved	Reserved																															
0x0A0	DTS_TSCSMPL _CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_DISCARD	SMPL_CTR_HOLD	SMPL_CTR_DISABLE
	Reset value																														0	0	0
0x0A4	DTS_TSCSDIF SMPLCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_CNTER_CLEAR
	Reset value																																0
0x0A8	DTS_TSCSMPL CNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_COUNT[15:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0 - 0x0BF	Reserved	Reserved																															

Table 334. DTS TS individual register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C0	DTS_TS0_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_EN_ALARMB	IRQ_EN_ALARMA	Res.	IRQ_EN_DONE	IRQ_EN_FAULT
	Reset value																												0	0		0	0
0x0C4	DTS_TS0_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_STATUS_ALARMB	IRQ_STATUS_ALARMA	Res.	IRQ_STATUS_DONE	IRQ_STATUS_FAULT
	Reset value																												0	0		0	0
0x0C8	DTS_TS0_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_CLEAR_ALARMB	IRQ_CLEAR_ALARMA	Res.	IRQ_CLEAR_DONE	IRQ_CLEAR_FAULT
	Reset value																												0	0		0	0
0x0CC	DTS_TS0IRQ TESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_TEST_ALARMB	IRQ_TEST_ALARMA	Res.	IRQ_TEST_DONE	IRQ_TEST_FAULT
	Reset value																												0	0		0	0
0x0D0	DTS_TSnSDIF RDATAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_RDATA[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4	DTS_TS0SDIF DONER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_SMPL_DONE
	Reset value																																0
0x0D8	DTS_TS0SDIF DATAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_FAULT	SAMPLE_TYPE	SAMPLE_DATA[15:0]															
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC - 0x0E8	Reserved	Reserved																															
0x0E0	DTS_TS0 ALARMA_ CFGR	HYSTA_THRESH[15:0]																ALARMA_THRESH[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 334. DTS TS individual register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0E4	DTS_TS0 ALARMB_ CFGR	HYSTB_THRESH[15:0]																ALARMB_THRESH[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E8	DTS_TS0HL SAMPLER	SMPL_HI[15:0]																SMPL_LO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0EC	DTS_TS0HILO RESETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_HI_CLR	SMPL_LO_CLR
	Reset value																															0	0

Refer to *Section 2.4* for the register boundary addresses.

38 Voltage reference buffer (VREFBUF)

38.1 VREFBUF introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs, and also as voltage reference for external components through the VREF+ pin.

38.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

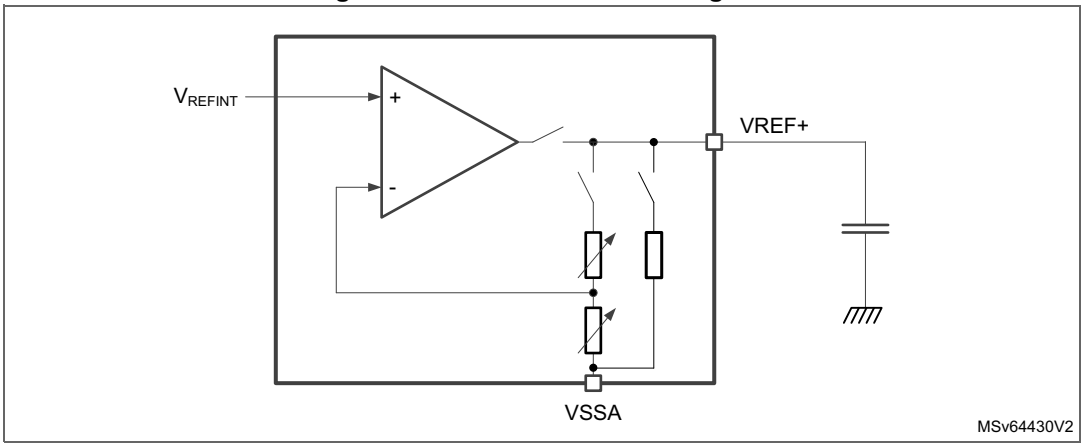
Table 335. VREFBUF typical values

Symbol	Value
VREFBUF0	1.21 V
VREFBUF1	1.5 V

Note: Refer to the product datasheet for more details.

38.3 VREFBUF functional description

Figure 297. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports four voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected
- VRS = 001: VREFBUF1 voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 336. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

38.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

38.5 VREFBUF power sequence

While VREFBUF is activated, V_{DDA} supply changing from 1.8 V to 0 V can cause large current flowing from the V_{REF+} output capacitor to the V_{DDA} power domain through on-chip parasitic resistance. To avoid this current, the following procedure is required:

- While V_{DDA} is available, configure VREFBUF in buffer off mode.
- Wait for V_{REF+} to discharge (through internal pull-down) to less than 100 mV (typically 1 ms, depending on the capacitor value).
- Switch off V_{DDA} supply to 0 V.

38.6 VREFBUF registers

38.6.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.

VRS = 001: VREFBUF1 voltage selected.

Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.

0: V_{REF+} pin is internally connected to the voltage reference buffer output.

1: V_{REF+} pin is high impedance.

Refer to *Table 336: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

38.6.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the OTP memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the OTP memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

38.6.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 337. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

39 Multi-function digital filter (MDF)

39.1 MDF introduction

The multi-function digital filter (MDF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- motor control
- metering

The MDF features 4 digital serial interfaces (SITFx) and digital filters (DFLTx) with flexible digital processing options to offer up to 24-bit final resolution.

The DFLTx of the MDF also include the filters of the audio digital filter (ADF).

The MDF can receive, via its serial interfaces, streams coming from various digital sensors.

The MDF supports the following standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

A flexible bitstream matrix (BSMX) allows the connection of any incoming bitstream to any filter.

The MDF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition it is possible to insert a high-pass filter or a DC offset correction block.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, decimation ratio, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible trigger interface can be used to control the conversion start. This timing control can trigger simultaneous conversions or insert a programmable delay between conversions.

The MDF features an out-off limit detectors (OLD) function. There is one OLD for each digital filter chain. Independent programmable thresholds are available for each OLD, making it very suitable for over-current detection.

A short circuit detector (SCD) is also available for every selected bitstream. The SCD is able to detect a short-circuit condition with a very short latency. Independent programmable thresholds are offered in order to define the short circuit condition.

The digital processing is performed using only the kernel clock. The MDF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

39.2 MDF main features

- AHB interface
- 4 serial digital inputs:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- Flexible matrix (BSMX) for connection between filters and digital inputs
- 2 inputs to connect the internal ADCs
- 4 flexible digital filter paths, including:
  - A configurable CIC filter:
    - Can be split into 2 CIC filters: high-resolution filter and out-off limit detector
    - Can be configured in $Sinc^4$ filter
    - Can be configured in $Sinc^5$ filter
    - Adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple
  - A high-pass filter to cancel the DC offset
  - An offset error cancellation
  - Gain control
  - Saturation blocks
  - An out-off limit detector
- Short-circuit detector
- Clock absence detector
- 16- or 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay independently each bitstream
- Various trigger possibilities
- Break generation on out-of limit or short-circuit detector events
- Autonomous functionality in Stop mode(s)
- DMA can be used to read the conversion data
- Interrupts services

39.3 MDF implementation

Table 338. MDF features (1)

Mode or feature	MDF1
Number of filters (DFLTx) and serial interfaces (SITFx)	4
MDF_CKI0 connected to pins	X
Sound activity detection (SAD)	-
RXFIFO depth (number of 24-bit words)	4
Number of trigger inputs	14

Table 338. MDF features (1) (continued)

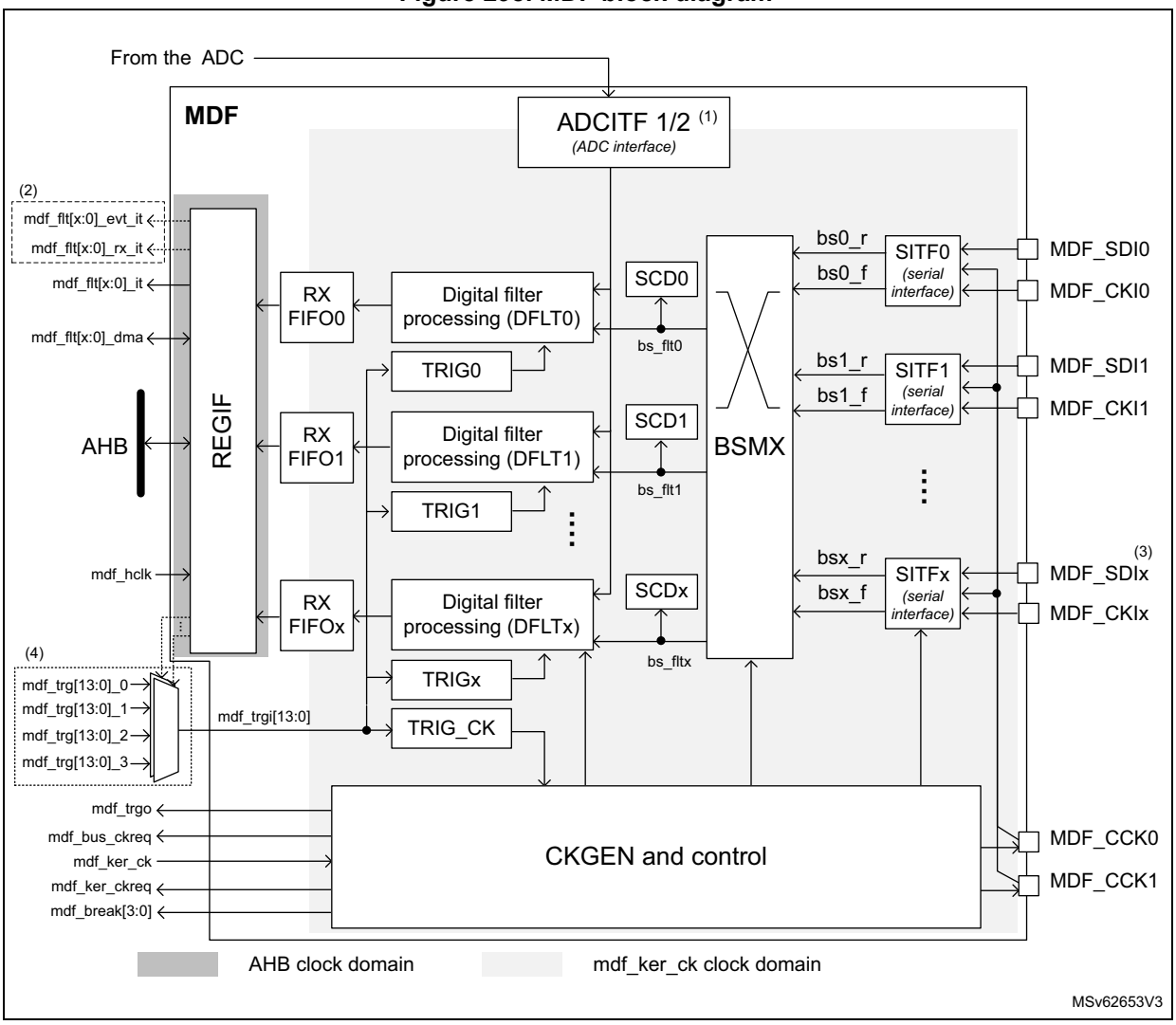
Mode or feature	MDF1
ADC connected to ADCITF1	ADC1
ADC connected to ADCITF2	ADC2
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	X
Main path with CIC4, CIC5	X
Main path with CIC1,2, 3 or FastSinc	X
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X
Global Interrupts MDF_FLT[x:0] (mdf_it[x:0])	X
Receive interrupts MDF_RX[x:0] (mdf_rx_it[x:0])	-
Event interrupts MDF_EVT[x:0] (mdf_evt_it[x:0])	-
Autonomous in Stop 0 and Stop 1 modes	X
Autonomous in Stop 2 mode	-
All DMA channels connected	X

1. 'X' = supported, '-' = not supported.

39.4 MDF functional description

39.4.1 MDF block diagram

Figure 298. MDF block diagram



1. Refer to *Section 39.3: MDF implementation* to check if the ADCITF is available, and which ADCs are connected.
2. Not always implemented, refer to the vector table of the product for details.
3. Refer to *Section 39.3: MDF implementation* to check the number of filter instances available.
4. The number of trigger inputs depends on the product. Refer to *Section 39.3: MDF implementation* for details.

39.4.2 MDF pins and internal signals

Table 339. MDF external pins

Name	Signal type	Comment
MDF_CKIy (y = 0 to 3)(1)	Input	Dedicated clock signals from external sensors
MDF_SDIy (y = 0 to 3)(1)	Input	Data signal from external sensors
MDF_CCKy (y = 0, 1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

1. The number of inputs available depends on the number of filters. Refer to *Section 39.3* for details.

Table 340. MDF internal signals

Name	Signal type	Comment
mdf_trgi[13:0]	Input	Trigger inputs in order to control the acquisition (see *Table 341: MDF trigger connections* for details)
mdf_trgo	Output	Trigger output for synchronizing several MDFs
mdf_break[3:0]	Output	Break signals event generation from over-current detector or short-circuit detector (see the next table for details)
mdf_flt[3:0]_dma(1)	Input/output	DMA request/acknowledge signals for each filter processing chain
mdf_flt[3:0]_it(1)	Output	Global interrupt signals, for each MDF filter
mdf_flt[3:0]_rx_it(1)(2)	Output	Receive interrupt signals, for each MDF filter.
mdf_flt[3:0]_evt_it(1)(2)	Output	Event interrupt signals, for each MDF filter.
mdf_bus_ckreq	Output	Bus interface clock request output
mdf_ker_ckreq	Output	Kernel clock request output
mdf_ker_ck	Input	Kernel clock input
mdf_hclk	Input	AHB bus interface clock input
mdf_adcitf1_dat[15:0]	Input	ADCITF1 data input
mdf_adcitf2_dat[15:0]	Input	ADCITF2 data input

1. The number of signals available depends on the number of filters. Refer to *Section 39.3* for details.
2. Not always connected. See the interrupt vector table for details.

The table below shows the way the trigger inputs of the MDF are connected.

Table 341. MDF trigger connections

Trigger name	Trigger source
mdf_trgi0	tim1_trgo2
mdf_trgi1	tim8_trgo2
mdf_trgi2	tim20_trgo2
mdf_trgi3	tim16_oc
mdf_trgi4	tim6_trgo

Table 341. MDF trigger connections (continued)

Trigger name	Trigger source
mdf_trgi5	tim7_trgo
mdf_trgi6	exti2_exti[11]
mdf_trgi7	exti2_exti[15]
mdf_trgi8	lptim1_ch1
mdf_trgi9	lptim2_ch1
mdf_trgi10	lptim3_ch1
mdf_trgi11	adf1_trgo
mdf_trgi12	adf1_sad_det (sound activity detection signal from ADF1)
mdf_trgi13	-

The table below shows the way the break outputs of the MDF are connected.

Table 342. MDF break connections

Trigger name	Trigger source
mdf_break0	tim1_break[8]
	tim15_break[8]
	tim20_break[8]
mdf_break1	tim1_break2[8]
	tim16_break[8]
	tim20_break2[8]
mdf_break2	tim8_break[8]
	tim17_break[8]
mdf_break3	tim8_break2[8]

Table 343. MDF ADC data connections

ADC data bus name	ADC source
mdf_adcitf1_dat[15:0]	adc1_dat
mdf_adcitf2_dat[15:0]	adc2_dat

39.4.3 Serial input interfaces (SITF)

The SITFx input interfaces allow the connection of the external sensors to the digital filters, via the bitstream matrix (BSMX). The SITFx serial interface can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode

- Manchester mode

The amount of SITFx instances is equal to the amount of filters.

The data from each serial interface can be routed to any filter in order to perform:

- the PDM to PCM conversion
- the out-off limit detection
- the short detection

The serial interfaces are enabled by setting the corresponding SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ΣΔ modulator.

Note: Before enabling the serial interface, the user must insure that the mdf_proc_ck is already enabled (see Section 39.4.5 for details).

The SITFx are controlled via the *MDF serial interface control register x (MDF_SITFxCR)*.

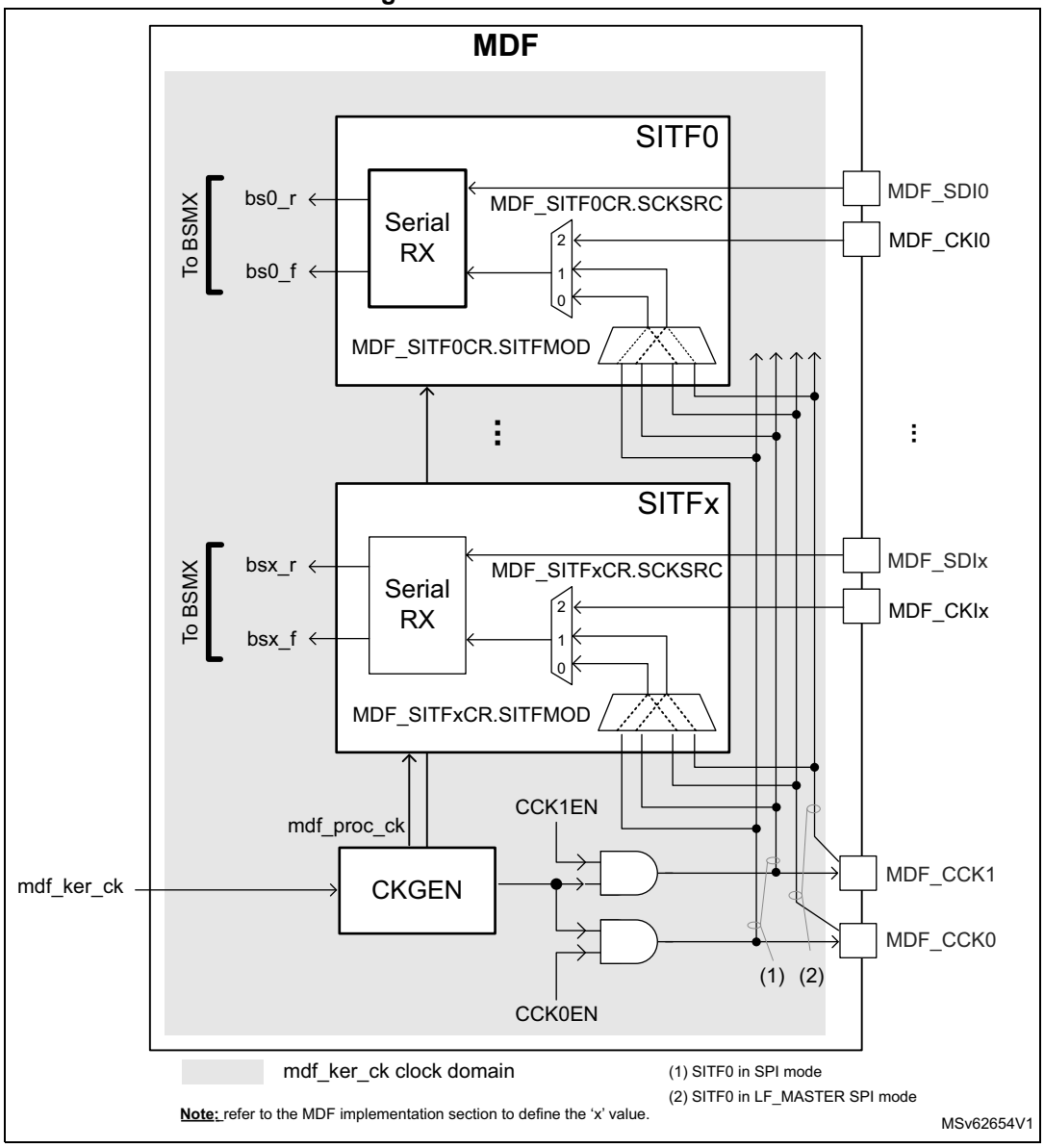
As shown in *Figure 299*, for each SITF, there is a large choice of clocking possibilities:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on MDF_CCK0 or MDF_CCK1 or MDF_CKIx pin (see 2 in *Figure 299*).
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source must be the clock directly provided by the CCKDIV to the MDF_CCK0 or MDF_CCK1 pin (see 1 in *Figure 299*). In this case MDF_CKIx must not be selected.

See *Table 344* for additional information.

Note: Using the common clock (MDF_CCK0 or MDF CCK1) can be helpful to share the same clock between several SITFx.

Figure 299. SITFx overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the usage of a mdf_proc_ck clock frequency, only two times higher than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the MDF must provide the bitstream clock to the external sensors via MDF_CCK0 and MDF_CCK1 pins, and receives the bitstream data via the serial data input MDF_SDIx.

For each SITFx, the application must select the same clock than the one provided to the external sensor (MDF_CCK0 or MDF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

Warning: The MDF_CKIx pin cannot be used in LF_MASTER SPI mode.

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the mdf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select MDF_CCK0, MDF_CCK1 or MDF_CKIx clock for the capture of the data received via the MDF_SDIx pin.

The MDF can generate a clock to the sensors via MDF_CCK0 or MDF_CCK1 if needed.

For all SPI modes, all SITFs can share the same clock input (MDF_CCK0 or MDF_CCK1), in order to optimize the amount of requested I/Os.

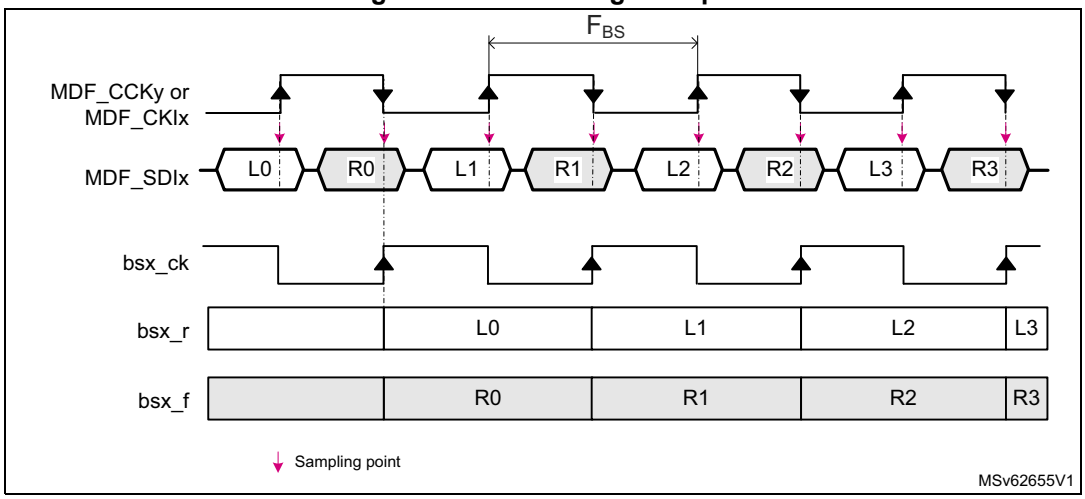
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITFx always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bsx_f)
- bitstream received using the bitstream clock rising edge (bsx_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 300. SPI timing example



To properly synchronize/receive the data stream, the frequency of the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 345*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the corresponding SITFx. If the MDF does not detect clock transitions for a duration of STH[4:0] x $T_{mdf_proc_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the *MDF serial interface control register x (MDF_SITFxCR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF i= 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

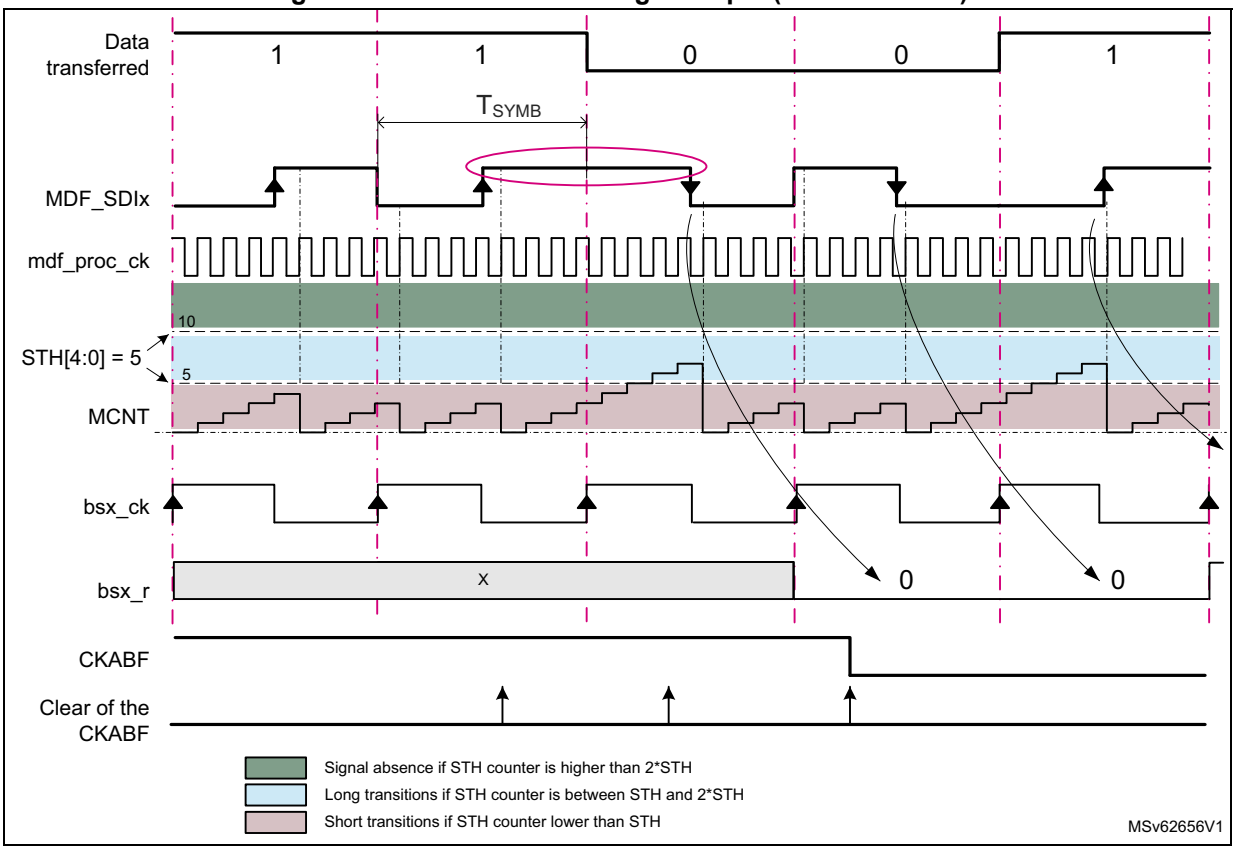
In Manchester coded format, the MDF receives data stream from the external sensor via the MDF_SDIx pin only.

The MDF_CKIx pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bsx_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 301. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program the STH[4:0] bitfield in the *MDF serial interface control register x (MDF_SITFxCR)*. The STH[4:0] bitfield is used by the SITFx to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the MDF_SDIx input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. *Figure 301* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1)\times T_{mdf_proc_ck} < T_{SYMB} < (2 \times STH \times T_{mdf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the mdf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 345*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{mdf_proc_ck}$, or when the SITFx is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the MDF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 301*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized, the hardware immediately sets the CKABF flag to 1.
2. Read the CKABF flag.
   – If CKABF = 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the MDF kernel clock frequency ($F_{mdf_ker_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}).

1. Provide a valid mdf_proc_ck to the SITFx.

   The mdf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{mdf_proc_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{mdf_proc_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = $\text{round}\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{mdf_proc_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{mdf_proc_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

39.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the MDF. Refer to *Section 39.3* to check the situation for this product.

The MDF allows the connection of up to two ADCs to the filter path. For each filter, the DATSRC[1:0] bitfield in the *MDF digital filter configuration register x (MDF_DFLTxCICR)* allows the application to select either data from the ADCs.

Warning: The MDF does not support receiving interleaved data from one of the ADCITF input.

39.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the MDF:

- AHB clock (mdf_hclk) used for the register interface
- kernel clock (mdf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the MDF_CCK0 and MDF_CCK1 pins. All those clocks are generated from the mdf_ker_ck.

The processing clock (mdf_proc_ck) is used to run all the signals processing and to re-sample the incoming serial or parallel stream.

Note: The reshape filter (RSFLT) needs up to 24 cycles of mdf_proc_ck clock to process a sample.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the MDF_CCK0 and MDF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

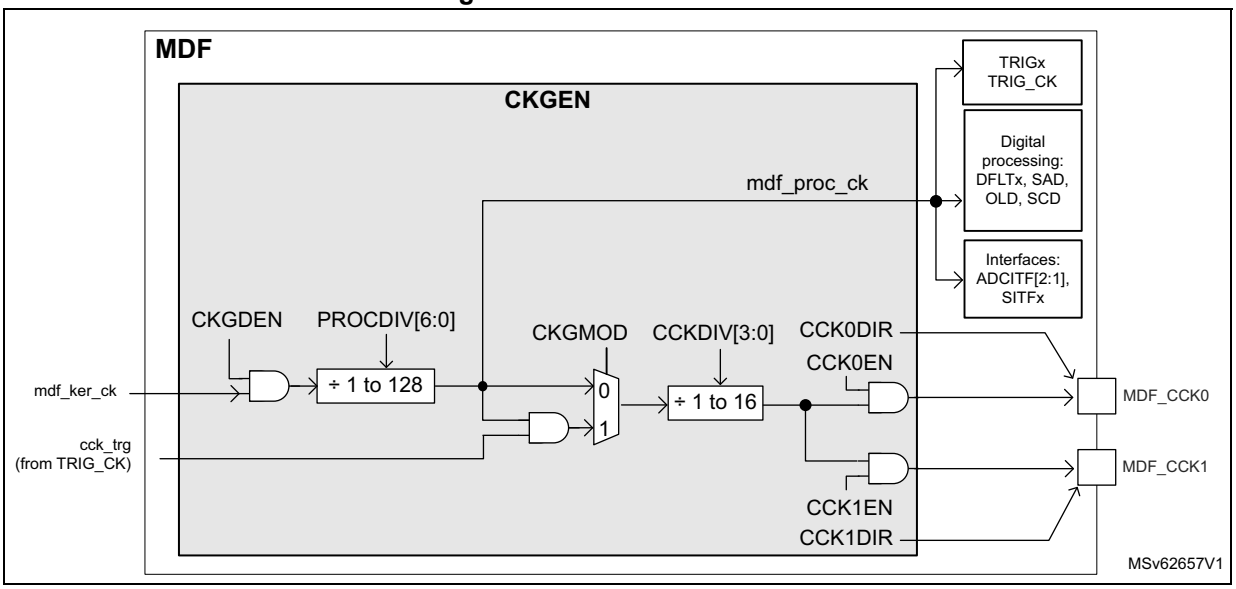
The mdf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All bits and fields controlling the CKGEN are in MDF_CKGCR.

Figure 302. CKGEN overview



The trigger logic for CKGEN is handled by the TRG_CK block. As shown in *Figure 311*, the CCKDIV divider can be triggered on the rising or falling edge of one of the 16 trigger sources. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the MDF_CCKy (y = 0,1) clock of several MDF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the MDF_CCK0 or MDF_CCK1 pin CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 344. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The MDF_CCKy pin is in input. An external clock can be connected to the MDF_CCKy pin and used by the SITFx in order to decode the serial stream
0	1	The MDF_CCKy pin is in output. No clock is generated, thus the MDF_CCKy pin is driven low.
1	1	The MDF_CCKy pin is in output. A clock is generated on the MDF_CCKy pin. The SITFx can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCKyDIR = 0 must be avoided (no interest).

When CCKyDIR = 1, as soon as the CCKyEN bit is set to 1, a clock is generated to the corresponding output without any spurs.

Note: The mdf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITFx or DFLTx).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0] and TRGSENS.
4. Set CKGDEN to 1.
5. Set CCKxDIR to 1 (optional).
6. Set CCKxEN to 1 (optional).

When needed, at any moment, the CCK0EN or CCK1EN value can be changed without disabling the clock generator.

Clock frequency constraints

The table below shows the frequency constraints to receive and process properly the samples.

Table 345. Clock constraints with respect to the incoming stream(1)

SITFx mode	MDF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{MDF_CCKy} max frequency limited to 5 MHz	
	$F_{mdf_proc_ck} > 2 * F_{MDF_CCKy}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$	$F_{mdf_proc_ck} > 24 * F_{MDF_CCKy} / (MCICD+1)$ and $F_{mdf_proc_ck} > 2 * F_{MDF_CCKy}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$
MASTER SPI SLAVE SPI	F_{MDF_CKx} max frequency limited to 25 MHz	
	$F_{mdf_proc_ck} > 4 * F_{MDF_CKKy}$ and F_{mdf_hclk} higher or equal to $F_{mdf_proc_ck}$	$F_{mdf_proc_ck} > 24 * F_{MDF_CKKy} / (MCICD+1)$ and $F_{mdf_proc_ck}$ higher than 4 x F_{MDF_CKKy} and F_{mdf_hclk} higher or equal to $F_{mdf_proc_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{mdf_proc_ck}$ higher than 6 * F_{SYMB} and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$	$F_{mdf_proc_ck}$ higher than 24 * $F_{MDF_CKKy} / (MCICD+1)$ and $F_{mdf_proc_ck} > 6 * F_{SYMB}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$

1. F_{MDF_CKKy} represents the frequency of the clock received via MDF_CKIx and MDF_CCKy, or generated via MDF_CCKy.F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

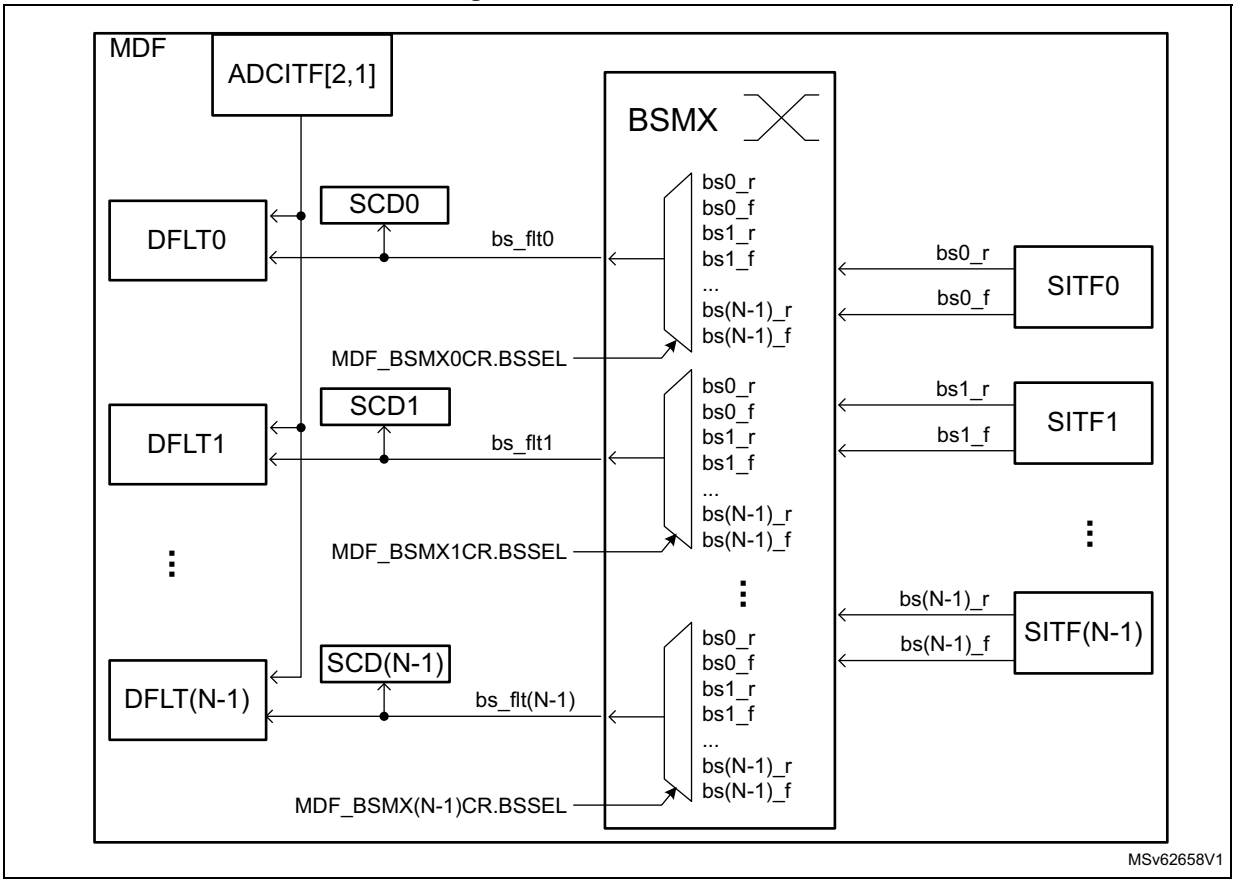
39.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from all serial interfaces (SITFx) and provides the selected input to the digital filters (DFLTx) and to the short-circuit detectors (SCDx). For each filter path, any of the bitstream input can be selected.

As shown in the *Figure 299*, each SITFx block provides two bitstreams (bsx_r and bsx_f) to the BSMX.

The application to select the wanted stream via the *MDF bitstream matrix control register x (MDF_BSMXxCR)*. This selection is intended to be static.

Figure 303. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the corresponding SCDx, OLDx or DFLTx is enabled. The following steps are needed to change the value of BSMX, for filter path x:

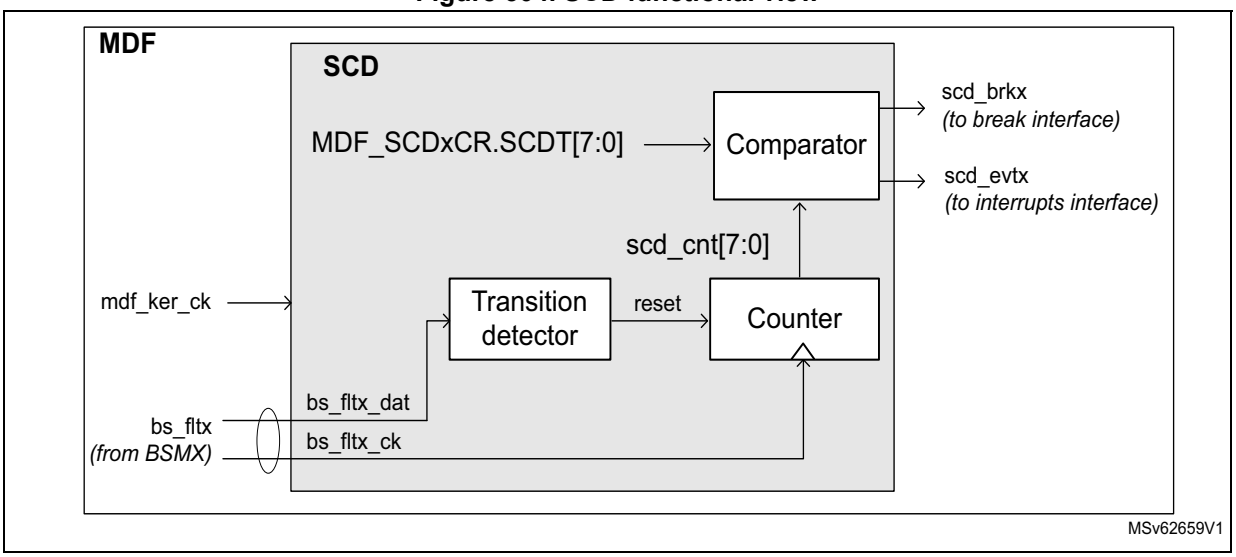
1. Set SCDEN of SCDx to 0.
2. Set DFLTEN of DFLTx to 0.
3. Set OLDEN of OLDx to 0.
4. Wait for BSMXACTIVE = 0.
5. Program BSSEL[4:0] of filter path x.
6. Set SCDEN of SCDx to 1 (optional).
7. Set DFLTEN of DFLTx to 1 (optional).
8. Set OLDEN of OLDx block to 1 (optional).

39.4.7 Short-circuit detectors (SCD)

The SCDx detects, with a very fast response time, if an analog signal reached saturated values (out-of-full scale ranges) and remained on this value for a given time.

This behavior can detect short-circuit or open-circuit errors (such as over current or over voltage). An interrupt event or/and a break signal can be generated.

Figure 304. SCD functional view

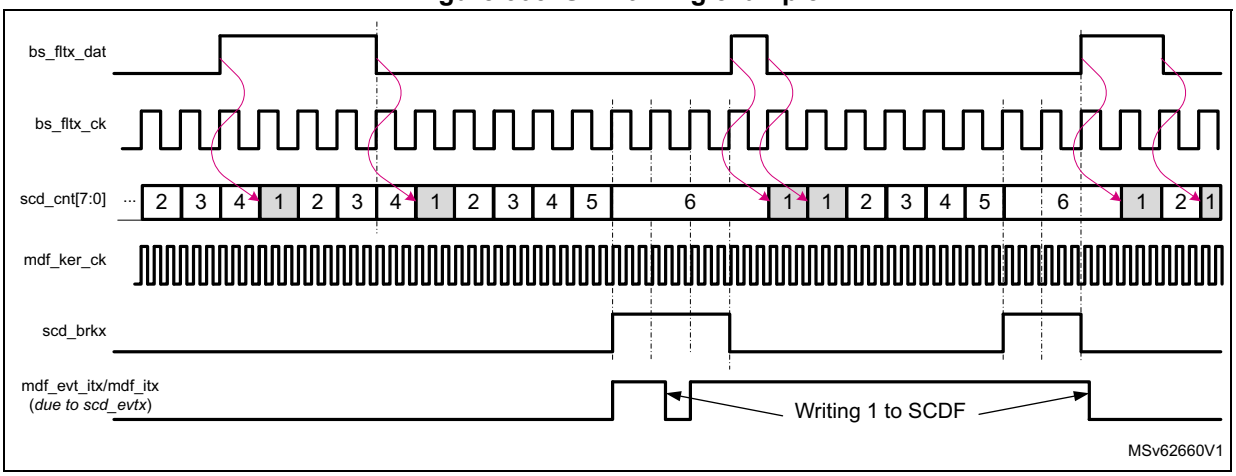


The SCDx is counting consecutive zeros or consecutive ones received from the serial interface (SITFx). A counter is restarted if there is a change in the data stream received. If this counter reaches a short-circuit threshold value (SCDT[7:0] in the *MDF short circuit detector control register x (MDF_SCDxCR)*), then a short-circuit event is invoked. Each BSMX output has its own short-circuit detector. Any BSMX output can be continuously monitored by setting the corresponding SCDEN bit to 1 in MDF_SCDxCR. Each SCD has its own threshold settings (SCDT) and its own status bit (SCDF).

The figure below shows an example where SCDT[7:0] is set to six. The break signal remains high as long as the short circuit condition is present.

No overrun event is generated when the interrupt event is pending while a new short-circuit condition is detected.

Figure 305. SCD timing example



The SCDx event generated by the SCDx block can be assigned to break output signals mdf_break[3:0]. The break signal assignment to a given short-circuit detector event is done by BKSCD[3:0] in MDF_SCDxCR. The break outputs are shared with the over-current function.

Note: SCDs cannot be used to monitor the ADC data interface (ADCITF).

SCD activation sequence example

1. Enable and configure CKGEN to generate the mdf_proc_ck.
2. Set SCDEN to 0.
3. Wait for SCDACTIVE = 0. If SCDEN was previously enabled, this phase can take two periods of mdf_hclk, and two periods of mdf_proc_ck.
4. Program BKSCD[3:0] and SCDT[7:0].
5. Set SCDEN to 1.

Note: BKSCD[3:0] and SCDT[7:0] must not be changed when SCDACTIVE = 1.

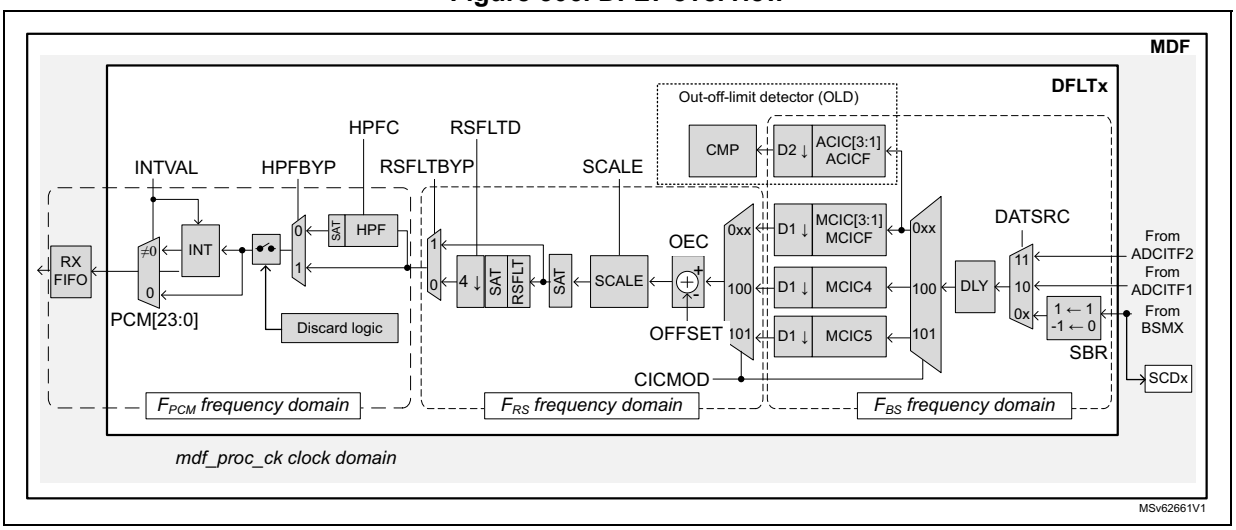
39.4.8 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- source selector
- clock skipper delay (DLY)
- CIC decimation filter that can be configured:
  - in single $Sinc^4$ or $Sinc^5$ order CIC (MCIC)
  - in two CIC filters:
    - a main filter (MCIC), high resolution
    - an auxiliary filter (ACIC), low-latency, for the out-off-limit detector (OLD)
    - both MCIC and ACIC can be configured as $Sinc^1$, $Sinc^2$, $Sinc^3$ or FastSinc
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- integrator (INT)
- receive RXFIFO

Figure 306 shows the filter-path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 306. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

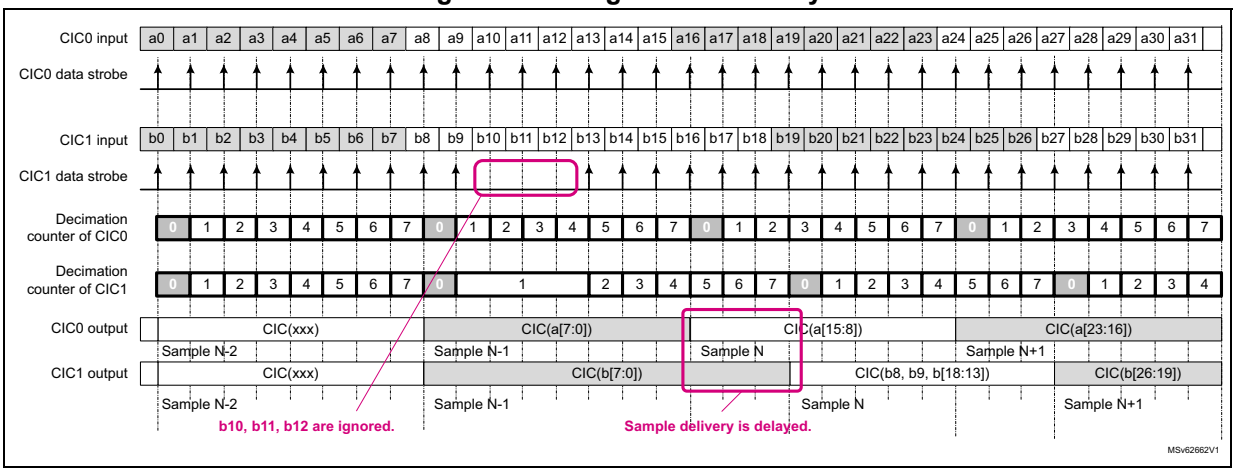
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate, must be applied to the incoming stream.This feature is helpful when the MDF is synchronized with other ADF or MDF blocks for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is performed by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

Figure 307 shows an example of how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another ADF or MDF instance.

Figure 307. Programmable delay



The CIC of filter 1 (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is build from input samples a[23:16] while the sample N+1 of CIC1 is build from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

Warning: It is not recommended to apply a delay bigger than the programmed decimation ratio (CIC + RSFLT decimation), especially if the MDF is programmed in interleaved acquisition mode. There is a strong risk of data misalignment inside the FIFOs.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the corresponding DFLTx is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the corresponding digital filter is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the MDF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the MDF is enabled. In a given filter, if the application performs a D1 delay, followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Data coming from ADCs can also benefit of this feature.

In interleaved acquisition mode, it is up to the application to insure that the delay applied on the different microphones is in line with the depth of the RXFIFO buffers. If the relative delay between each activated filter is less than the decimation ratio, then it costs one FIFO location.

If the interleaved acquisition mode is not used, then the delay value has no impact on the RXFIFO buffering.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The $Sinc^N$ digital filter embedded into the MDF is configurable according to the application targeted.

- For audio applications, such as speech capture from digital microphones, the application can select a high-resolution low-pass decimation filter by setting CICMOD to 100 or 101.
- If the targeted application is motor control or any other sensor capture, then the application can configure the CIC in order to offer a main filter path for acquisition (MCIC) and an auxiliary filter for fast-event generation (ACIC). This auxiliary filter can be used for example, to detect over-current conditions. This mode is selected by setting CICMOD[2:0] = 0xx.

When CICMOD[2:0] = 0xx, the following CIC filters are available:

- main filter (MCIC)
- auxiliary CIC filter (ACIC)

Both of them are configurable in FastSinc, $Sinc^1$ to $Sinc^3$.

When CICMOD[2:0] = 100, the CIC is configured into a single $Sinc^4$ and when CICMOD[2:0] = 101, the CIC is configured into a single $Sinc^5$ filter.

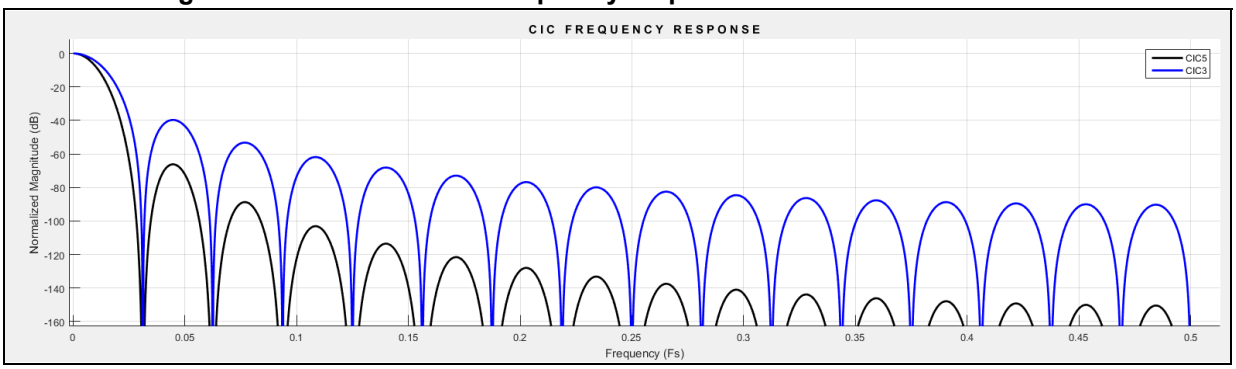
The filters have the following transfer function (impulse response in z domain):

- $Sinc^N$ filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$
- FastSinc filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot D)})$$

where N can be 1, 2, 3, 4 or 5, and D is the decimation ratio.

D is equal to MCICD+1 or ACICD+1.

Figure 308. CIC3 and CIC5 frequency response with decimation ratio = 32



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}) depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^{N}$$

The decimation ratio can be adjusted from 2 to 512 for the main CIC filter and from 1 to 32 for the auxiliary CIC filter.

The table below gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream but can be up to 16 bits when coming from the ADCITF.

Table 346. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITFx)						Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)					
	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
4	3	5	6	7	9	11	14	16	17	18	20	22
8	4	7	8	10	13	16	15	18	19	21	24	-
12	5	9	9	12	16	19	16	20	21	23	-	-
16	5	9	10	13	17	21	16	20	21	24	-	-
24	6	11	12	15	20	24	17	22	23	-	-	-
32	6	11	12	16	21	26	17	22	23	-	-	-

Table 346. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITFx)						Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)					
	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
64	7	13	14	19	25	-	18	24	25	-	-	-
128	8	15	16	22	-	-	19	26	-	-	-	-
256	9	17	18	25	-	-	20	-	-	-	-	-
512	10	19	20	-	-	-	21	-	-	-	-	-

The LSB part of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off-band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

The table below shows the maximum allowed decimation ratio for the CIC filter, depending on the input data size. Bigger decimation ratio causes a wrap-around of the signal at CIC output, for strong input signals.

Note: The MDF cannot detect or prevent a CIC wrap-around.

Table 347. Maximum decimation ratio versus order and input data size

Filter order	Max. decimation ratio when DS_{IN} = 1 bit (SITFx)	Max. decimation ratio when DS_{IN} = 12 bits (ADCITF)	Max. decimation ratio when DS_{IN} = 16 bits (ADCITF)
Sinc^1	512	512	512
Sinc^2	512	128	32
FastSinc	512	90	22
Sinc^3	322	25	10
Sinc^4	76	11	5
Sinc^5	32	6	4

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB) and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] bitfield in the *MDF digital filter configuration register x (MDF_DFLTxCICR)*.

SCALE[5:0] can be changed even if the corresponding DLFTx is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

The table below shows the possible gain values.

Table 348. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The MDF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block). The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Gain adjustment policy

To get the best MDF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF

– a maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048) for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

The SCALE value in dB is selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^N}\right)$$

where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

Example: for a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7:

– If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^4}\right) \sim 11 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 11 dB. The closest lower value is - 12dB (SCALE[5:0] = 0x2C).

– If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^4}\right) \sim 23 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

The table below proposes gain values for different filter configurations, when the data comes from the SITFx, according to the CIC order, and the CIC decimation ratio. This table is not exhaustive, and considers a full-scale input signal (see *Section 39.7.5* for details).

Table 349. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)					Gain settings (dB) for configuration SITF + CICx (+ HPF)				
	CIC5	CIC4	CIC3	CIC2	CIC1	CIC5	CIC4	CIC3	CIC2	CIC1
8	33.6	51.7	69.7	72.2	72.2	45.7	63.7	72.2	72.2	72.2
12	18.1	39.6	60.2			30.1	51.7			
16	3.5	27.6	51.7			15.6	39.6	63.7		
20	- 6.0	21.6	48.2			6.0	33.6	60.2		
24	- 12.0	15.6	42.1	69.7		0	27.6	54.2		
28	- 20.6	9.5	36.1	66.2		- 8.5	21.6	48.2		
32	-26.6	3.5	33.6	63.7		- 14.5	15.6	45.7		
48	-	- 8.5	24.1	57.7		-	3.5	31.6	69.7	
64	-	- 20.6	15.6	51.7		-	-8.5	27.6	63.7	
128	-	-	- 2.5	39.6		-	-	9.5	51.7	
256	-	-	- 20.6	27.6		-	-	-8.5	39.6	

Reshaping filter (RSFLT)

In addition to the CIC, the MDF offers a reshaping IIR filter mainly dedicated to the audio application but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in MDF_DFLTxRSFR.

The table below shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

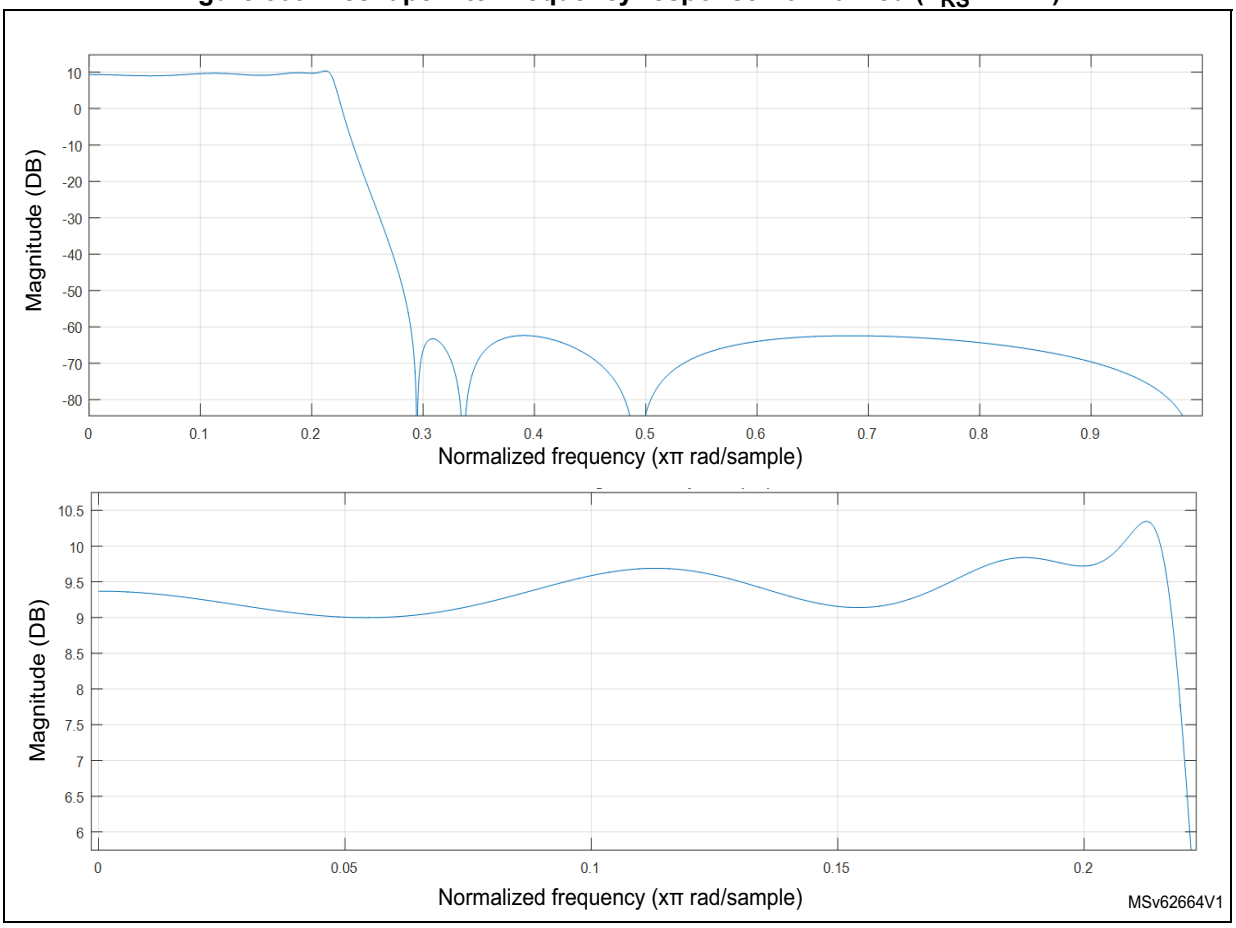
$$F_C = 0.111 \times F_{RS}$$

Table 350. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

The figure below shows the frequency response of the reshape filter.

Figure 309. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is about 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of mdf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the mdf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new sample while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provide x a faster mdf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source and it must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 351. HPF 3 dB cut-off frequencies examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

Offset error compensation (OEC)

Each digital filter has its own OEC. The offset correction is performed by subtracting to the signal provided by the CIC, the OFFSET[25:0] in the *MDF offset error compensation control register x (MDF_OECxCR)*.

Due to the internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new offset, and the moment where the new offset value is effectively applied to the samples. If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back OFFSET[25:0] informs the application on the current offset value.

Integrator (INT)

The INT performs additional decimation and a resolution increase of data coming from the digital filter. The INT simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The INT is enabled by setting INTVAL[6:0] to a value different from 0.

The amount of integrated values can be defined by INTVAL[6:0] in the range of 2 to 128.

In order to control the data width at the integrator output, the resulting data can be divided by 1, 4, 32 or 128. This feature is controlled by INTDIV[1:0].

39.4.9 Out-of-limit detector (OLD)

The OLD triggers an event when a signal reaches or crosses given maximum and minimum threshold values. The generated event can drive an interrupt or break signals (mdf_break[3:0]) when conditions are met.

The OLD can be used only if the CIC filter is configured in motor and sensing mode (CICMOD[2:0] = 0xx).

The OLD takes the input signal selected by the main filter, process it using a $Sinc^N$ or FastSinc filter (ACIC), and compares the resulting signal to programmed thresholds.

The OLD is enabled via the OLDEN bit in MDF_OLD0CR. Once enabled, the input data are continuously monitored. There is no need to have the DFLTx enabled for using the OLD function.

The MDF offers a high- and low-threshold register that are compared with given data values.

The application can generate an event if the signal is inside or outside the boundary defined by those two thresholds. This behavior is controlled via THINB in the MDF_OLD0CR.

If the application only wants to generate an event when the input signal is higher than OLDTHH, then THINB and OLDTHL must both be cleared to 0.

If the application only wants to generate an event when the input signal is lower than OLDTHH, then THINB must be set to 1 and OLDTHL must be cleared to 0.

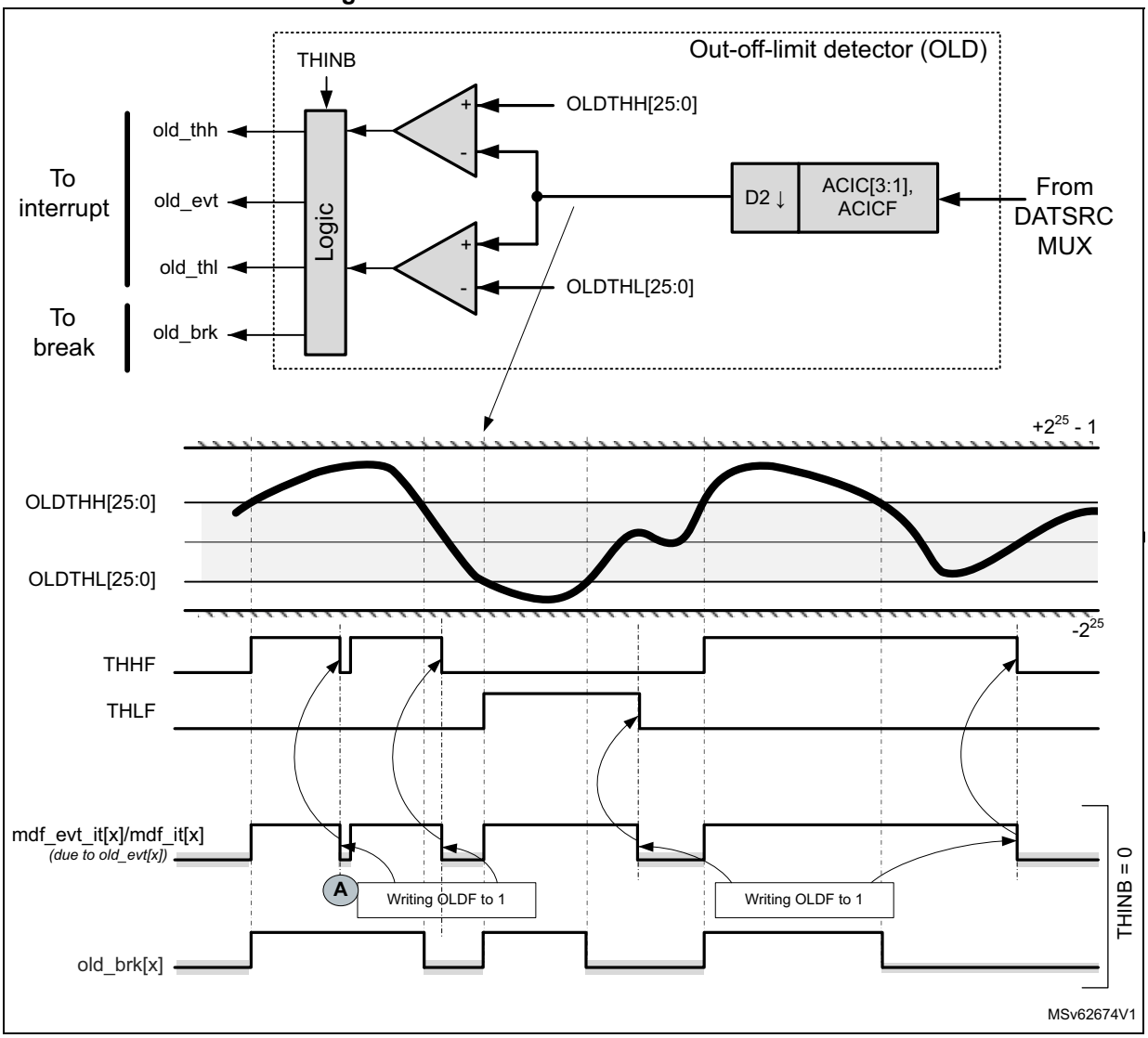
Note: It is not recommended to set a OLDTHL to a value bigger than OLDTHH.

The response time of the OLD depends on several parameters, the most important are listed hereafter:

- sampling rate frequency used by the external sensor
- ACIC decimation ratio
- ACIC order

The OLD can be used for over-current detection but also as current limiter if the PWM signal is generated by a timer receiving a old_brk signal. Generally, to get a fast response time for over-current detection, it is recommended to use the ACIC with lowest order as possible and the minimum decimation ratio. FastSinc is also a good choice for over-current detection. The application must perform a trade-off between the filter resolution and the response time.

Figure 310. Out-of-limit detector thresholds



Status flags are available in order to inform the application that an OLD event is detected. Latched events are cleared by writing 1 into the OLDF bit.

As shown in *Figure 310*, when THINB = 0, the interrupt signal remains active as long as the signal is outside the gray area. At position A, the application clears the interrupt, but the interrupt is re-asserted because the signal is still outside the gray area. When the application clears OLDF, THHF and THLF are cleared as well.

An OLD event can be assigned to one or several break output signals (mdf_break[3:0]). The break signal assignment to a given OLD event is done by BKOLD[3:0] in the *MDF out-of limit detector control register x (MDF_OLDxCR)*.

Note: The generation of break signals is independent from the interrupts generation.

OLD activation sequence example

1. Enable and configure CKGEN.
2. Set OLDEN to 0.

3. Wait for OLDACTIVE = 0. If OLDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
4. Program BKOLD[3:0], ACICN[1:0], ACICD[4:0], THINB,OLDTHL[25:0] and OLDTHH[25:0].
5. Set OLDEN to 1.

39.4.10 Digital filter acquisition modes

The MDF offers the following modes to perform a data capture:

- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode
- synchronous snapshot acquisition mode

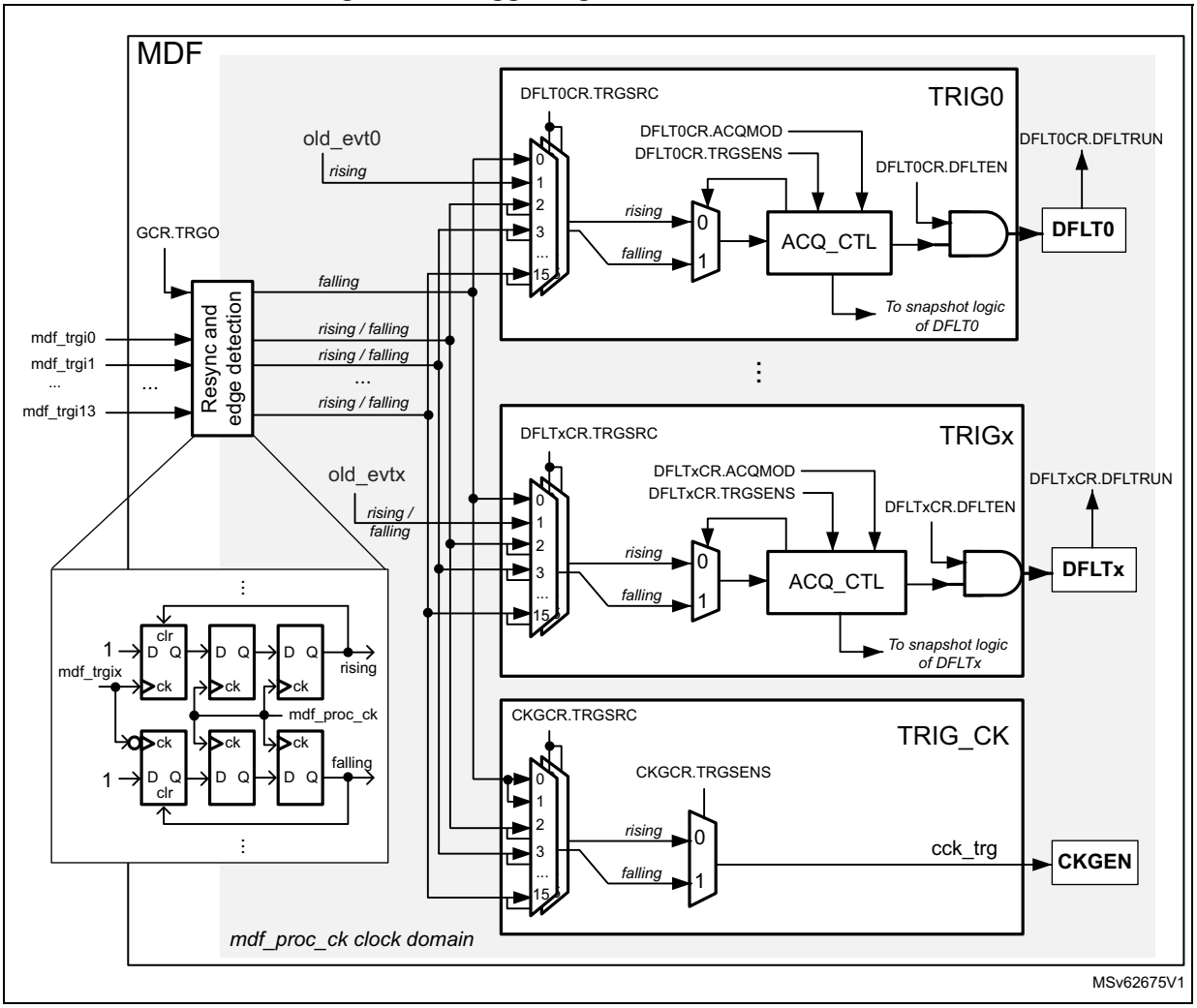
For each filter, one of these modes can be selected independently.

Note: To perform a data capture, the filters, the interfaces providing the data (SITFx or ADCITF) and the CKGEN must be enabled. If needed, MDF_CCK0 or MDF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This resets the filters and flushes the RXFIFO of the corresponding filter path. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filters are reset.

The figure below shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 311. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the resynchronization of the input triggers to the mdf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the mdf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select one of the following trigger sources:

- mdf_trgi[13:0] signals (refer to *Table 341* for the trigger connections)
- TRGO bit in MDF_GCR
- OLD event of each filter path

The edge sensitivity can also be selected, except for TRGO and OLD events.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by simply writing their DFLTEN bits to 1.

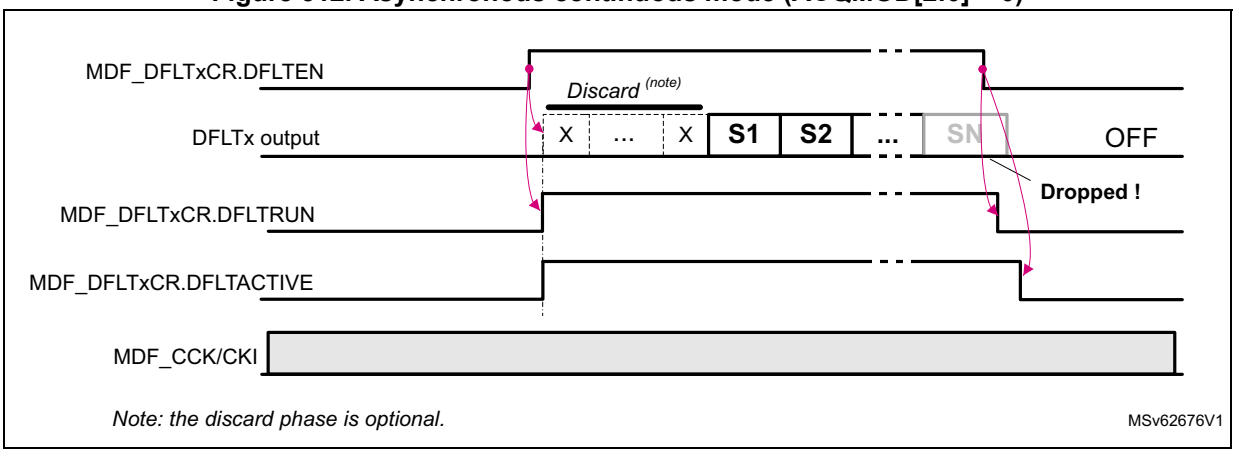
The Asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN) so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 355*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, the acquisition sequence starts immediately.

The figure below shows a simplified example of the samples generated by the DFLTx.

Figure 312. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample on one or several filters by simply writing their DFLTEN bits to 1.

The Asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are cleared to 0):

1. Configure and enable the clock generator (CKGEN), so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 351*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit of the requested data interfaces.

5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, each selected filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, for each filter, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

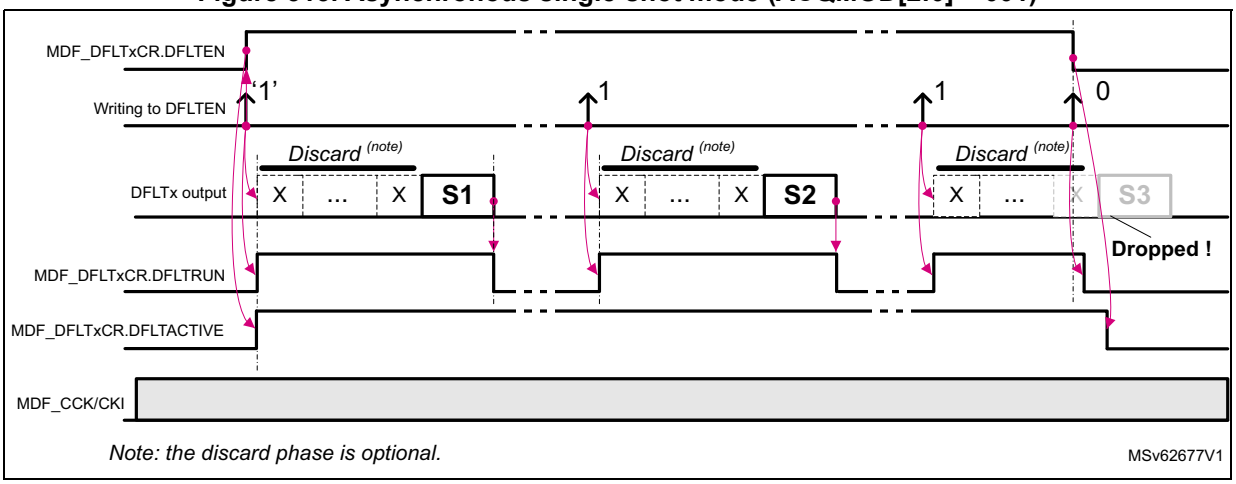
This sequence can be repeated every time a new data must be converted.

As shown in the *Figure 313*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If the DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 313. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

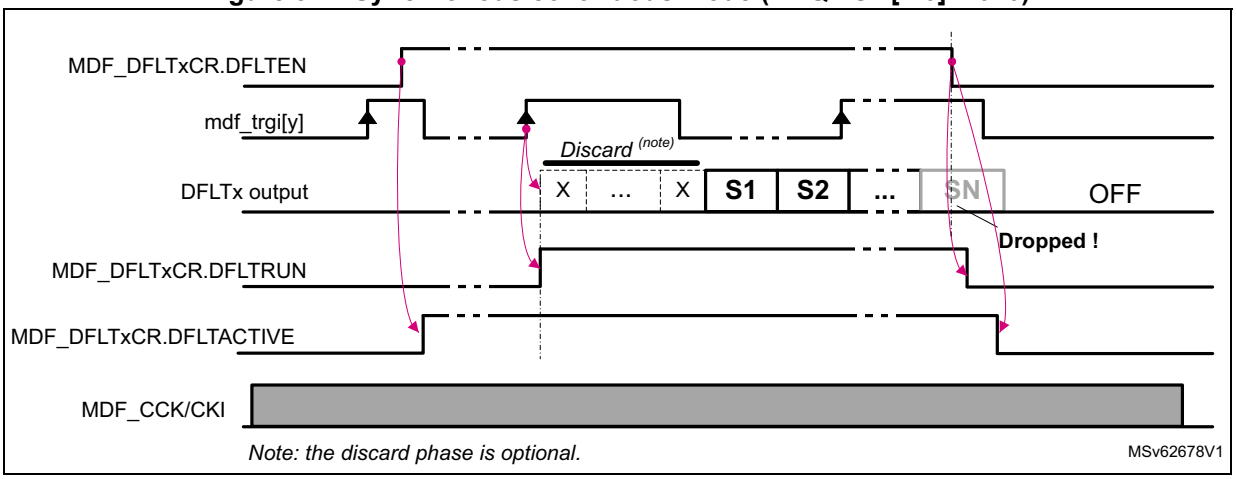
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 351*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 010.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1 for the filters to enable.
8. When the trigger condition is met, the filters start the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. The next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 314. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

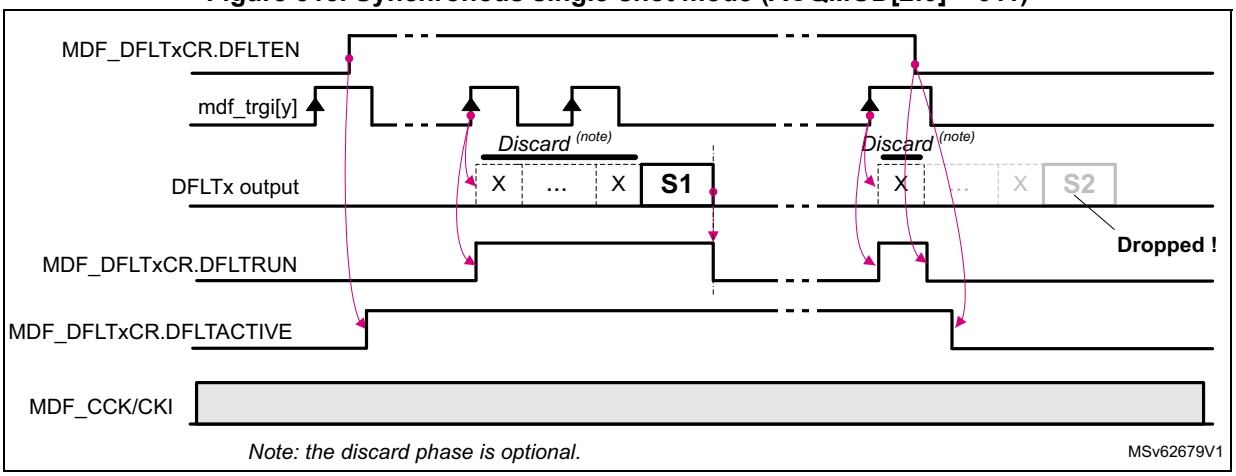
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 351*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter configuration and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. Then when the trigger condition is met, the filters start the acquisition and provide one data to the RXFIFO, then the filters are ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 315. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 315 shows a case where DFLTEN is cleared to 0 while an acquisition is ongoing: thus the sample S2 is lost. This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Note: The ongoing DMA transfer is properly terminated.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition on one or several filters controlled by consecutive edges of one of the following trigger sources:

- one of the mdf_trg[13:0] signals
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

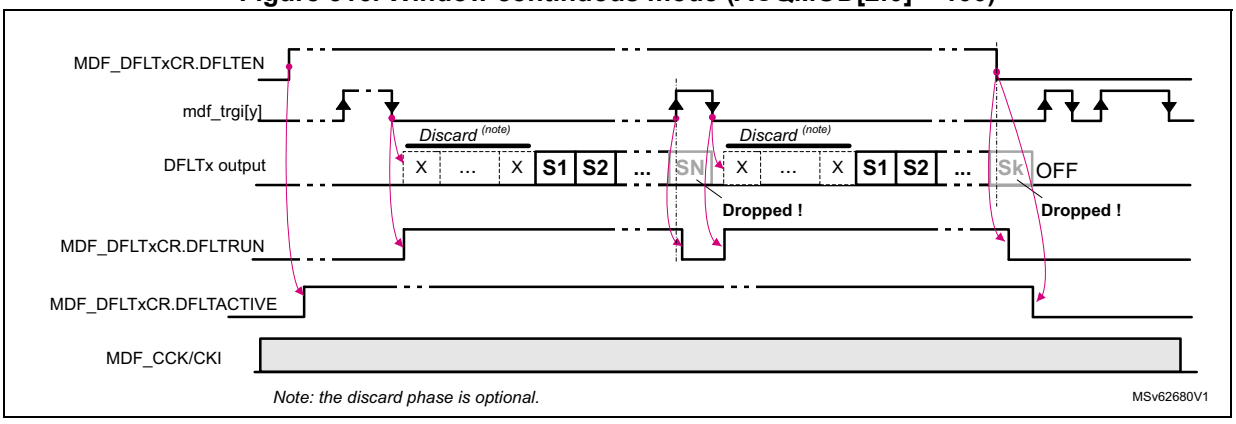
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 351*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 316 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the MDF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the MDF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the MDF waits again for a falling edge on the selected trigger input.

Figure 316. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA

transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous snapshot acquisition mode

In the acquisition modes described on previous sections, the application must wait for filters settlings time or the end of conversion before getting a new valid sample. For applications very critical in latency, the last valid sample can be get immediately, with an information on the age of this sample by using the snapshot acquisition mode.

In snapshot mode, the DFLTx is continuously acquiring the samples, but the processed samples are not stored into the RXFIFO. When a trigger occurs, the last valid sample, the current decimation value of MCIC and the sample counter of the INT are stored in MDF_SNPS0DR.

The possible trigger sources are the following:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous snapshot acquisition mode is selected when ACQMOD[2:0] = 101.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 351*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 101. FTH is usually cleared to 0 to receive an interrupt as soon as the RXFIFO is not empty.
4. Set the SITFEN bit of the requested data interfaces to 1.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated,
7. Set DFLTEN to 1 for the filters to enable.

When the trigger condition is met,

1. The MDF stores the last valid data provided by the corresponding filter, the value of the MCIC decimation counter, and the counter value of INT if selected by SNPSFMT into a shadow register.
2. The snps_evt is activated, indicating that a data is ready. If SSDRIE is set to 1, the MDF requests the AHB clock.
3. When the AHB clock is available the shadow register is stored in *MDF snapshot data register x (MDF_SNPSxDR)* and an interrupt is generated.
4. In the interrupt sub-routine, the application must do the following:
   a) Read the data located in MDF_SNPSxDR.
   b) Clear the SSDRF flag by writing it to 1.

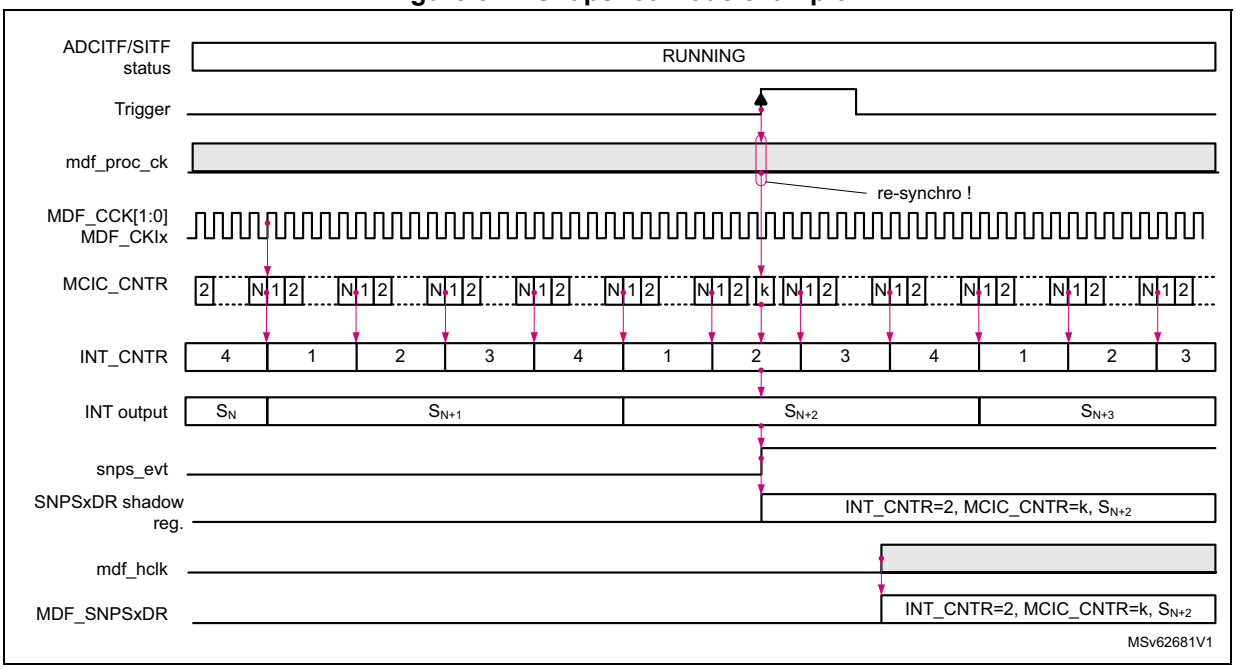
As shown in the *Figure 317*, in snapshot mode, when the trigger event occurs, it is resynchronized with the processing clock (mdf_proc_ck), the decimation counter of the MCIC (MCIC_CNTR), the integrator counter (INT_CNTR) and the last valid sample (S_{N+2}), are stored into a shadow register, and the snps_evt event is generated. When the AHB

clock is available, an interrupt is generated and the shadow register is copied in MDF_SNPSxDR. The application can read immediately the sample and the timing information.

The SSOVRF overrun flag is also available, allowing the application to check if an overrun condition occurs. A snapshot overrun condition is detected when the SSDRF flag is set to 1 while a new trigger event occurs.

In that case, the new trigger event is not taken into account. The application must clear the SSOVRF flag.

Figure 317. Snapshot mode example



Note: When an activated filter did not receive a data strobe, the filter is frozen. If the interface providing the stream is not enabled (SITF or ADCITF), the filter is frozen. Once a DFLT x is activated, CIC, HPF, IIR and INT are reinitialized. For each filter a status bit indicates if the filter is currently running or not.

Starting several filters synchronously

To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN is set to 0):

1. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
2. Set the SITFEN bit of the requested data interfaces to 1.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] to 0 (TRGO is selected).
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot), the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again TRGO to 1.

Discarded samples

The MDF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

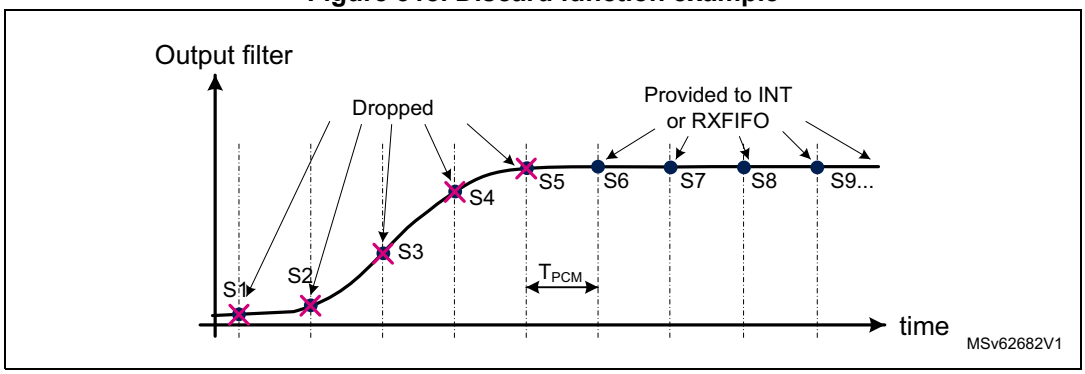
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 312* to *Figure 316*, *Figure 319* and *Figure 320*.

In the example shown in the figure below, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO (or INT block if enabled) is S6.

Figure 318. Discard function example



Warning: All filters working in interleaved DMA mode must have the same NBDIS[7:0] value.

Data interface activation

The data interfaces are enabled by setting the corresponding SITFEN or ADCITFEN bit to 1. Once the digital filter is enabled, it receives the serial data from the external ΣΔ modulator or parallel internal data sources (ADC).

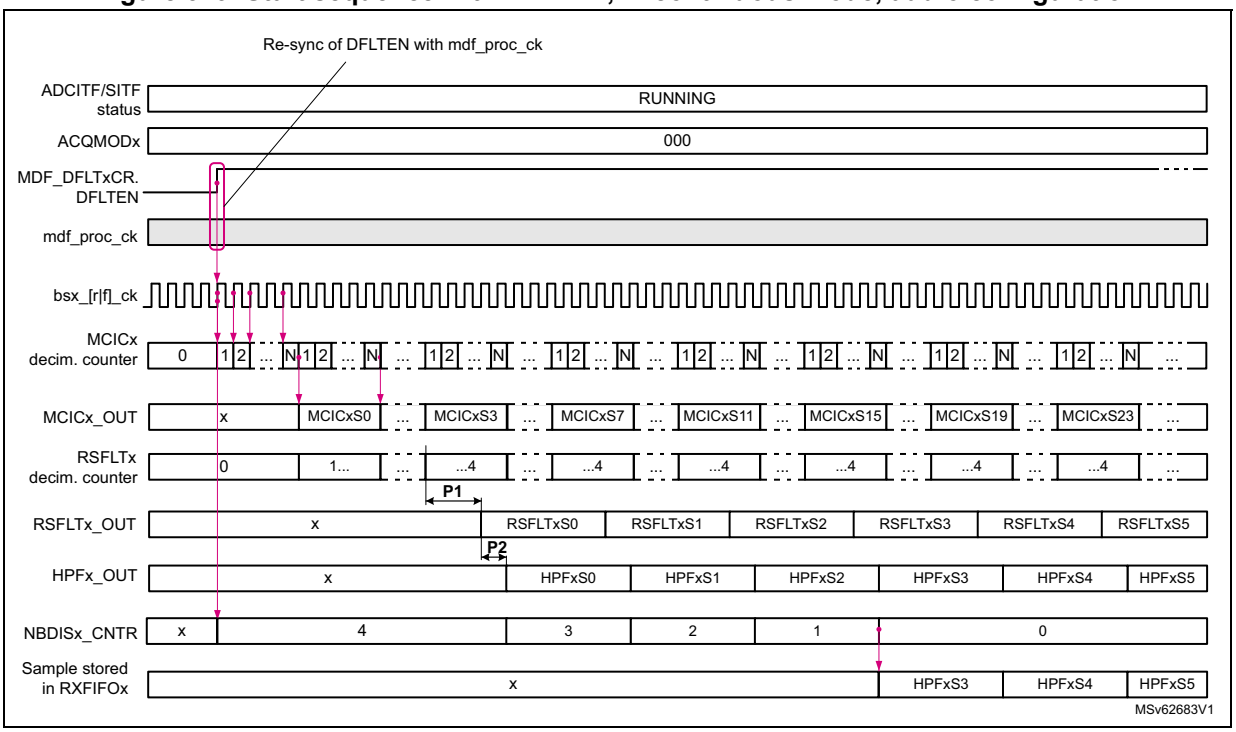
39.4.11 Start-up sequence examples

Figure 319 details an acquisition sequence start of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLTx is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITFx or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a low value to simplify the drawing.

Figure 319. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the MDF processing clock domain. When DFLTEN is detected high, the filter chain is enabled and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC and delivers a sample to the HPF every time it processes four samples (decimation by 4) The RSFLT needs up to 24 cycles of mdf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the HPF delivers a sample.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

The example shown in *Figure 320* is based in a motor-control filter configuration:

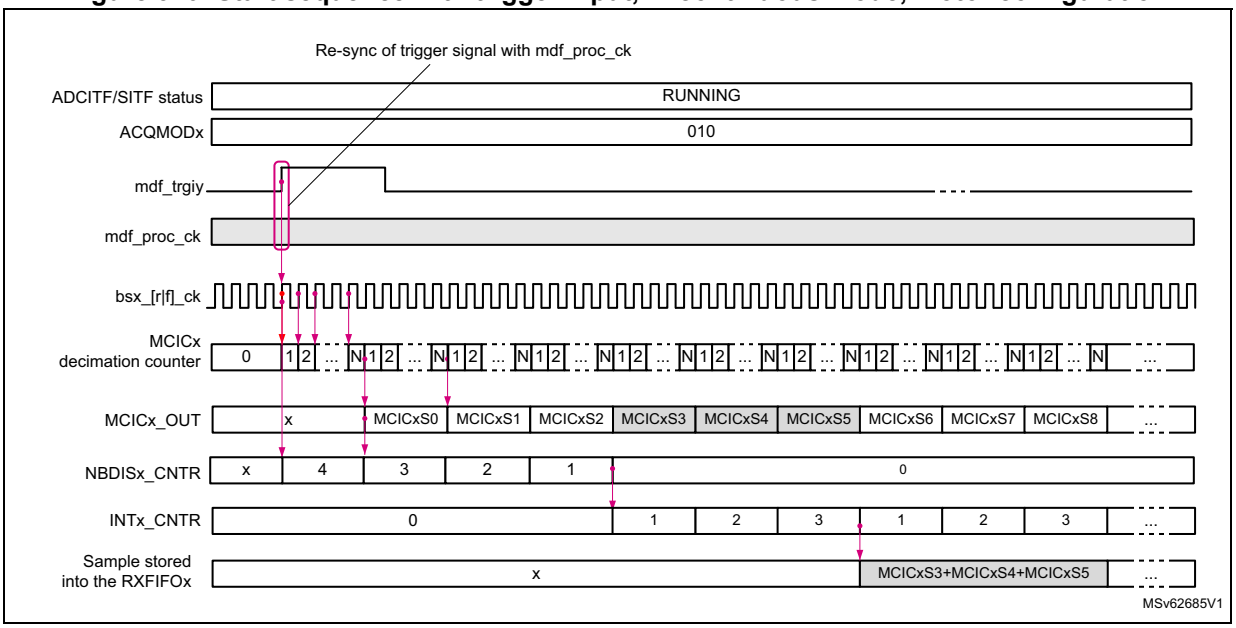
- MCIC and INT are enabled.
- SFLT and HPF are disabled.
- ACQMOD[2:0] is set to 1 and uses a mdf_trgiy input as trigger.
- NBDIS[7:0] is set to 3.
- INTVAL[6:0] is set to 2.

The mdf_trgiy input signal is re-sampled into the MDF processing clock domain to avoid any metastability issues. When the trigger condition is met (rising edge), the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available. The NBDIS function prevents the samples writing in the INT as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the MCIC delivers a sample.

When NBDIS_CNTR reaches 0, the new samples are provided to the INT, that performs the integration of three consecutive samples and stores them into the RXFIFO.

Figure 320. Start sequence with trigger input, in continuous mode, motor configuration

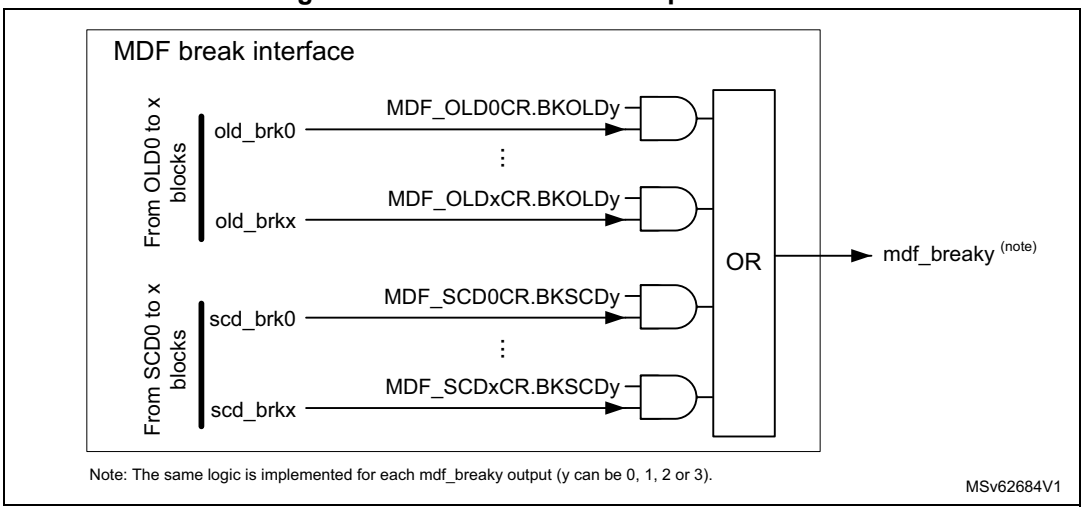


39.4.12 Break interface

The break interface merges the break events coming from the OLDx and SDCx blocks into four break signals connected to various peripherals of the product (see *Table 341* for details).

As shown in *Figure 321*, several blocks can share the same break line. A same block can also drive several break lines. The break interface is controlled via MDF_OLDxCR and MDF_SCDxCR.

Figure 321. Break interface simplified view



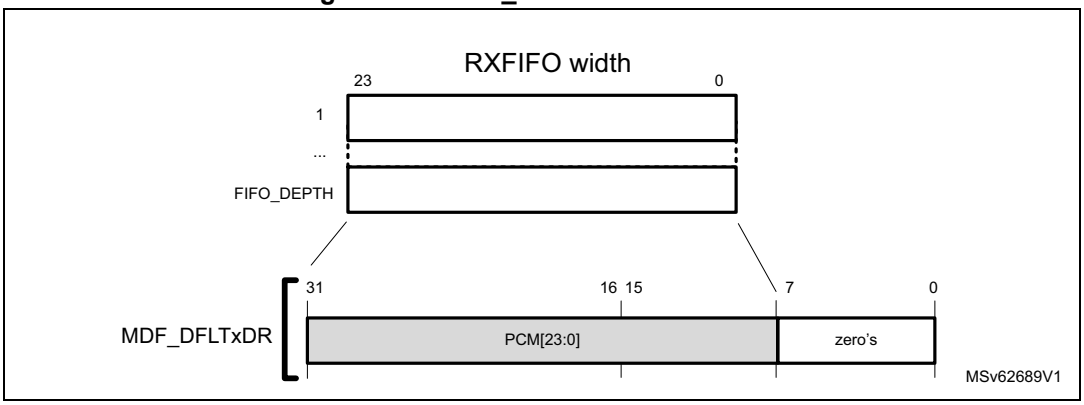
39.4.13 Data transfer to memory

Data format

The samples processed by DFLTx are stored into a RXFIFO. The application can read the samples stored into the FIFOs via the *MDF digital filter data register x (MDF_DFLTxDR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The MDF provides 24-bit left-aligned data. Performing a 16-bit access to MDF_DFLTxDR allows the application to get the 16 most significant bits. Performing a 32-bit access to MDF_DFLTxDR allows the application to get a 24-bit data size.

Figure 322. MDF_DFLTxDR data format



Data re-synchronization

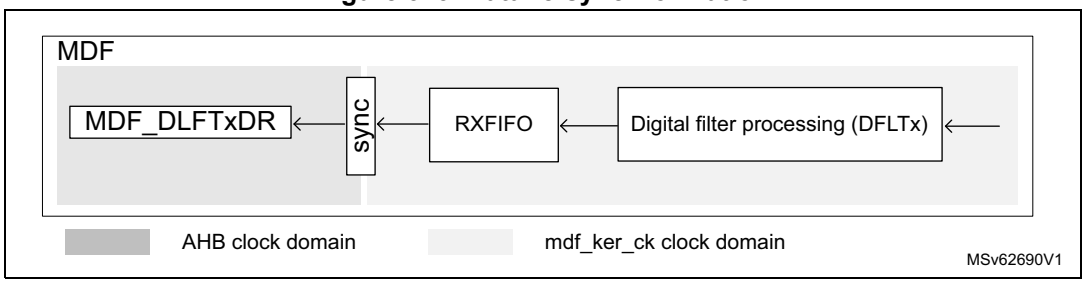
The samples stored into the RXFIFOs can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFOs are located into the mdf_ker_ck clock domain, while MDF_DFLTxDR *are located into the mdf_hclk (AHB) clock domain.*

When the AHB clock is available, if MDF_DFLTxDR is empty and if a sample is available into the RXFIFO, this sample is transferred into MDF_DFLTxDR.

The sample transfer from the RXFIFO to MDF_DFLTxDR takes two periods of the AHB clock (mdf_hclk) and two periods of the mdf_ker_ck clock. The MDF inserts automatically wait-states if the application performs a read operation of MDF_DFLTxDR while the transfer of the new sample from the RXFIFO to MDF_DFLTxDR is not yet completed.

Figure 323. Data re-synchronization



The MDF can also combine two transfer types: the independent-transfer and the interleaved-transfer modes.

Independent-transfer mode

In this mode, each RXFIFO has its own DMA channel and its own FTHF flag event.

Both single and burst DMA transfers are supported in this mode, but the application must care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data re-synchronization explained in the previous section.

In addition, the application can select the RXFIFO threshold (FTH bit) to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth / 2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the corresponding DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the corresponding RXFIFO is not empty.

The DMA mode of the RXFIFOx is enabled via the corresponding DMAEN bit in the *MDF digital filter control register x (MDF_DFLTxCR)*.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in MDF_DFLTxDR, the corresponding FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as MDF_DFLTxDR is read.
- If FTH = 1, as soon as one of the RXFIFO reaches the threshold level and a data is available into MDF_DFLTxDR, the corresponding FTHF is set, allowing the generation of an interrupt. The FTHF flag is released as soon as one data is read. FTHF is set again if the threshold condition is met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE / 2 data.

The independent-transfer mode must be used when the sample rates provided by each filter paths are not perfectly synchronous, or when the streams are independent. This situation may occur for example:

- when SITF is in SLAVE mode and each external sensor provides its own sampling clock

- when the decimation ratios of the DFLTx are not the same
- when all the streams are not starting at the same time

Interleaved-transfer mode

This mode optimizes the DMA request resources by sharing a single DMA request with several RXFIFOs.

Only single DMA transfers are supported in this mode.

In interleaved-transfer mode, FTH cannot be used and only the FTHF of RXFIFO0 is available for the interrupt signaling.

For the DMA transfer, when all the RXFIFOs working in interleaved-transfer mode are not empty, some DMA requests are generated in order to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR.

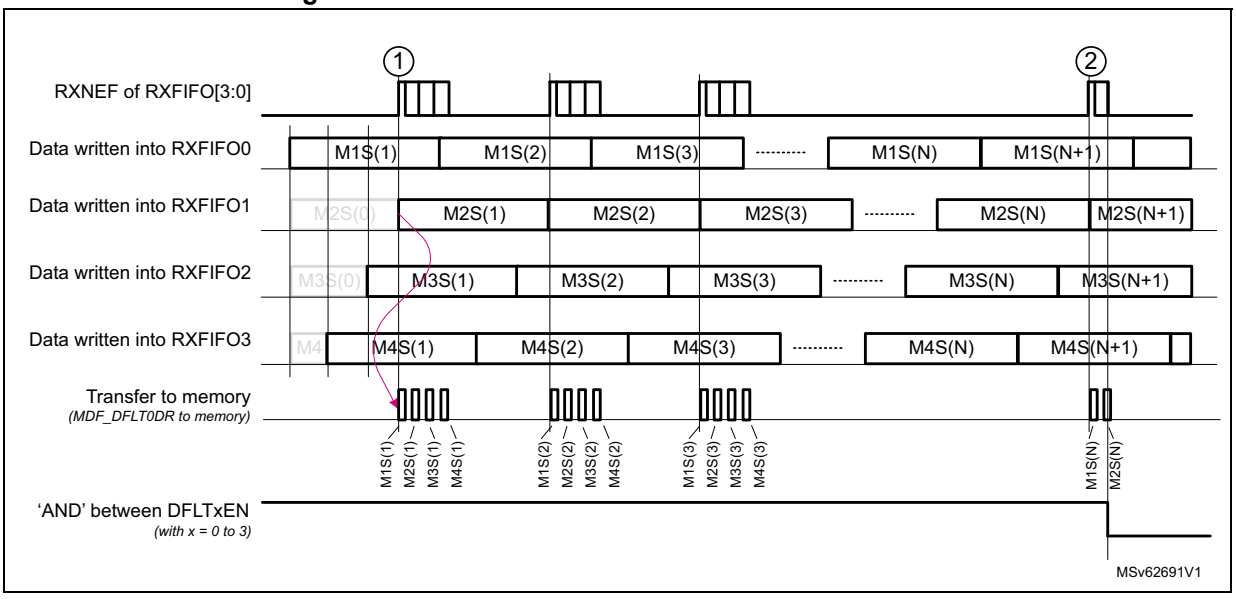
For the RXFIFOs working in interleaved-transfer mode, the DMA channel is enabled by setting to 1 DMAEN in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved mode via ILVNB[3:0]. DMAEN for the other RXFIFOs working in interleaved-transfer mode (RXFIFO 1 to ILVNB[3:0]) are not taken into account.

Figure 324 shows four RXFIFOs working in interleaved-transfer mode. Each RXFIFO has a delay due to the programming of the DLY block.

When all the RXFIFOs have at least one sample available (in this example the last FIFO receiving a sample is RXFIFO1), the MDF requests the data transfer to the memory (see 1 in *Figure 324*). In this figure, the DMA is assumed to be used for the data transfer to memory.

The acquisition can be simply stopped by setting DFTLEN to 0 for one of the filter configured in interleaved-transfer mode. In the example, when a DFLTEN is set to 0 (see 2 in *Figure 324*), the transfer to memory is immediately stopped. Samples M2S(N), M3S(N) and M4S(N) may be lost.

Figure 324. Data transfer in interleaved-transfer mode



In interrupt mode, when all the RXFIFOs working in interleaved-transfer mode have a data ready, the FTHF flag of RXFIFO0 is asserted to allow the interrupt generation. The application is then supposed to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR. The FTHF event is released when the first data is read.

For the RXFIFOs working in interleaved-transfer mode, the interrupt mode is enabled by setting FTHIE to 1 in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved-transfer mode via ILVNB[3:0]. FTHIE of the other RXFIFOs working in interleaved-transfer mode (RXFIFO1 to ILVNB[3:0]) are not taken into account.

The interleaved-transfer mode can only be applied starting at RXFIFO0 up to RXFIFOk. The number of RXFIFOs working in interleaved-transfer mode are defined by ILVNB[3:0].

When ILVNB[3:0] = 0, all RXFIFOs work in independent-transfer mode.

The interleaved-transfer mode can be used only for digital filters delivering samples perfectly synchronous each other. This is typically the case of sound capture with an array of digital microphones sharing the same bitstream clock.

Caution: To make the MDF working properly in interleaved-transfer mode, the following rules must be respected:

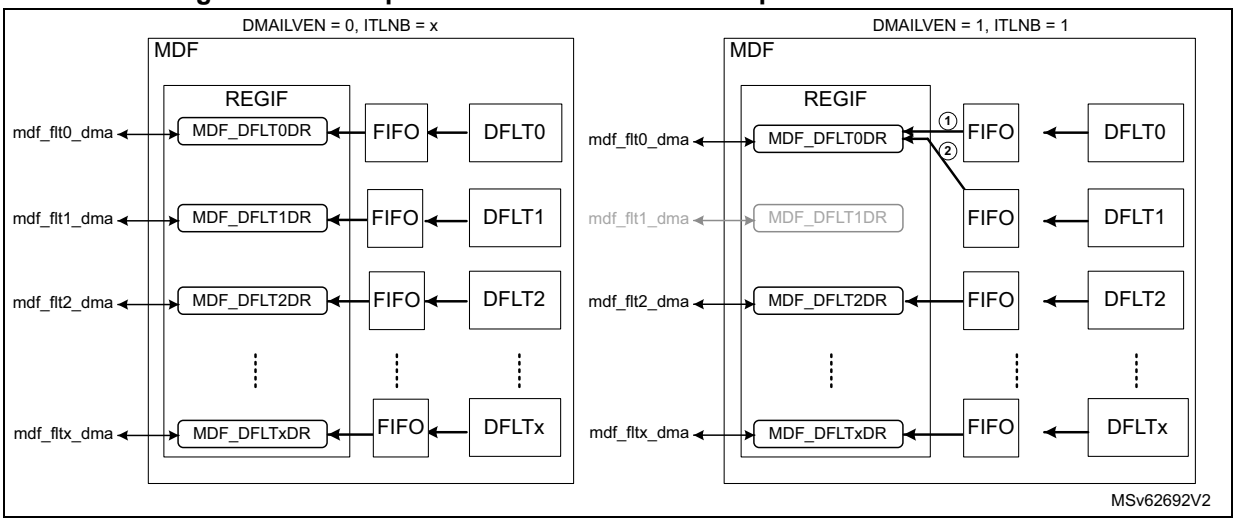
- All filters working in interleaved-transfer mode must be all synchronized together, meaning that all filters must select the same ACQMOD[2:0] (must be equal to 010 or 011), the same trigger source and the same trigger sensitivity.
- All filters working in interleaved-transfer mode must have the same configuration.
- All sensors providing the data for the filters working in interleaved-transfer mode must use the same bitstream clock frequency.
- The maximum delay difference applied with SPKDLY[6:0] for the filters working in interleaved-transfer mode must remain strictly lower than the RXFIFO depth. Normally this delay does not need to exceed one T_{PCM} period.

Note: Both independent- and interleaved-transfer modes can work in parallel: for example, the first RXFIFOs (RXFIFO0 to RXFIFOk) may work in interleaved-transfer mode when the others (RXFIFO(k+1) to RXFIFO(N)) works in independent-transfer mode.

If the AHB clock is not present when a data transfer must be performed, the MDF first requests the AHB clock (refer to Section 39.4.5: Clock generator (CKGEN) for details).

The figure below shows the data path for a configuration in independent-transfer mode (left-hand figure) and a configuration mixing both independent- and interleaved-transfer modes (right-hand figure).

Figure 325. Data path for interleaved- and independent-transfer modes



In the right-hand figure, the DFLT0 and DFLT1 filters are configured in interleaved-transfer mode. For those filters, FTH is no longer taken into account by the hardware. The samples provided by DFLT0 and DFLT1 are read from MDF_DFLT0DR. When both RXFIFOs are no longer empty, the MDF generates two DMA requests to read the sample from DFLT0 and the sample from DFLT1. If both RXFIFOs are not empty again, the same sequence is triggered.

Note: When one of the filter working in interleaved-transfer mode is disabled, the data transfer to memory of all filters in interleaved-transfer mode is stopped immediately as well.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full and a new sample from the DFLTx must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 326 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

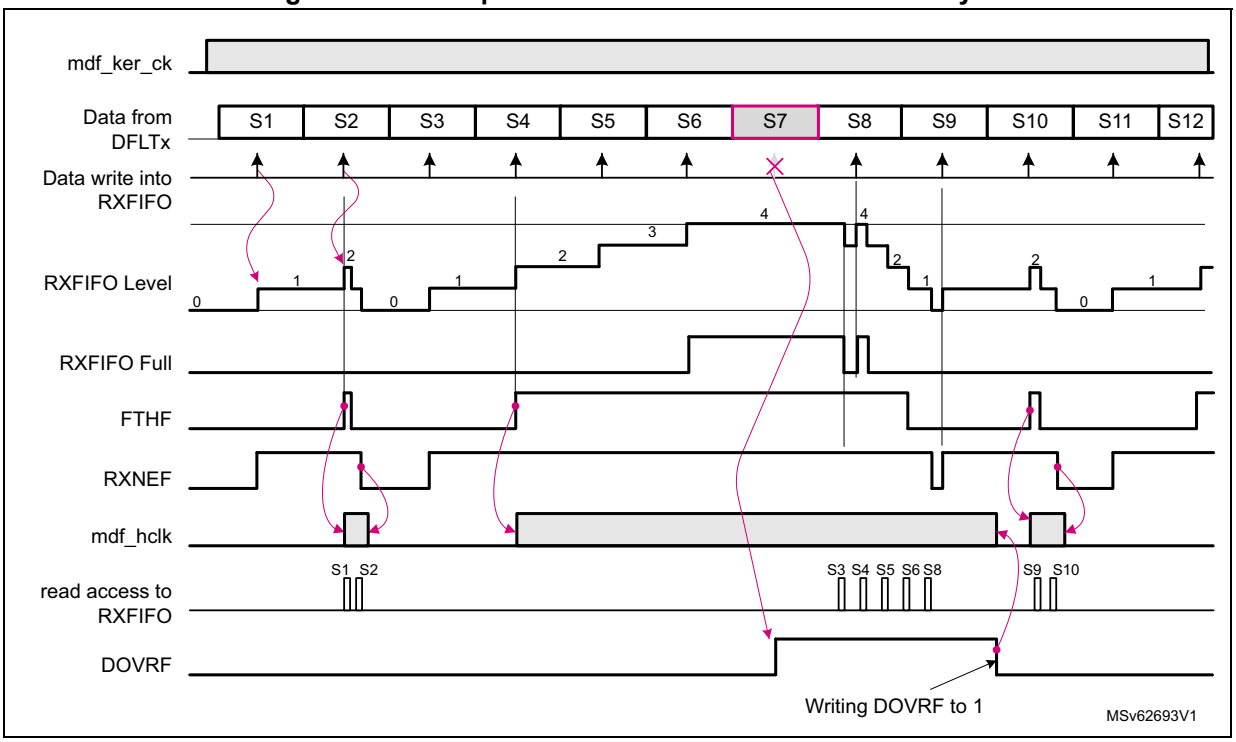
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO and the MDF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the mdf_hclk line, the gray boxes indicate that the MDF requested the AHB clock. The figure below shows the AHB clock available only when the MDF requests it. In real applications, the AHB clock may also be present if the MDF does not request it.

Figure 326. Example of overflow and transfer to memory



Note: If the MDF works in interleaved-transfer mode, the application must check the overrun status of all RXFIFOs that works in interleaved-transfer mode.

39.4.14 Autonomous mode

The MDF can work even if the AHB bus clock is not available (Stop modes). The MDF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the MDF receives a kernel clock if the selected clock source is an oscillator available in Stop mode

In Stop mode, the MDF receives the AHB clock if the following conditions are met:

- The MDF autonomous mode is enabled in the RCC.
- The MDF requests the AHB clock, in the following situations:
  - when the MDF must transfer data into the memory via the DMA

    The data are directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the MDF must generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt

flag is active. More precisely, when an interrupt must be generated, the MDF requests the AHB clock.

39.4.15 Register protection

The MDF embeds some hardware protection to prevent invalid situations. The table below shows the list of write-protected and unprotected fields.

Table 352. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF global control register (MDF_GCR)	TRGO	ILVNB[3:0]	DFLTACTIVE0 = 1
MDF clock generator control register (MDF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
MDF serial interface control register x (MDF_SITFxCR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
MDF bitstream matrix control register x (MDF_BSMXxCR)	-	BSSEL[4:0]	DFLTACTIVEx = 1 or SCDACTIVEx = 1 or OLDACTIVEx = 1
MDF digital filter control register x (MDF_DFLTxCR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
MDF digital filter configuration register x (MDF_DFLTxCICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
MDF reshape filter configuration register x (MDF_DFLTxRSFR)	-	All fields	
MDF integrator configuration register x (MDF_DFLTxINTR)	-	All fields	
MDF out-of limit detector control register x (MDF_OLDxCR)	OLDEN	ACICD[4:0], ACICN[1:0],THINB BKOLD[3:0]	OLDACTIVEx = 1
MDF OLDx low threshold register x (MDF_OLDxTHLR)	-	All fields	
MDF OLDx high threshold register x (MDF_OLDxTHHR)	-	All fields	
MDF delay control register x (MDF_DLYxCR)	-	SKPDLY[6:0]	SKPBF = 1
MDF short circuit detector control register x (MDF_SCDxCR)	SCDEN	BKSCD[3:0], SCDT[7:0]	SCDACTIVEx = 1
MDF DFLTx interrupt enable register x (MDF_DFLTxIER)	All fields	-	-

Table 352. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF DFLTx interrupt status register x (MDF_DFLTxISR)	All fields	-	-
MDF offset error compensation control register x (MDF_OECxCR)	OFFSET[25:0]	-	-

All the MDF processing is performed in the mdf_proc_ck clock domain. For that reason, enabling or disabling a MDF sub-block may take some time due to the re-synchronization between the AHB clock domain and the mdf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for more details.

39.5 MDF low-power modes

Table 353. Effect of low-power modes on MDF

Mode	Description
Sleep	No effect. MDF interrupts cause the device to exit Sleep mode.
Stop(1)	The MDF registers content is kept. If the MDF is clocked by an internal oscillator available in Stop mode, the MDF remains active. The interrupts cause the device to exit Stop mode.
Standby	The MDF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 39.3: MDF implementation* for details about Stop modes supported by the MDF.

39.6 MDF interrupts

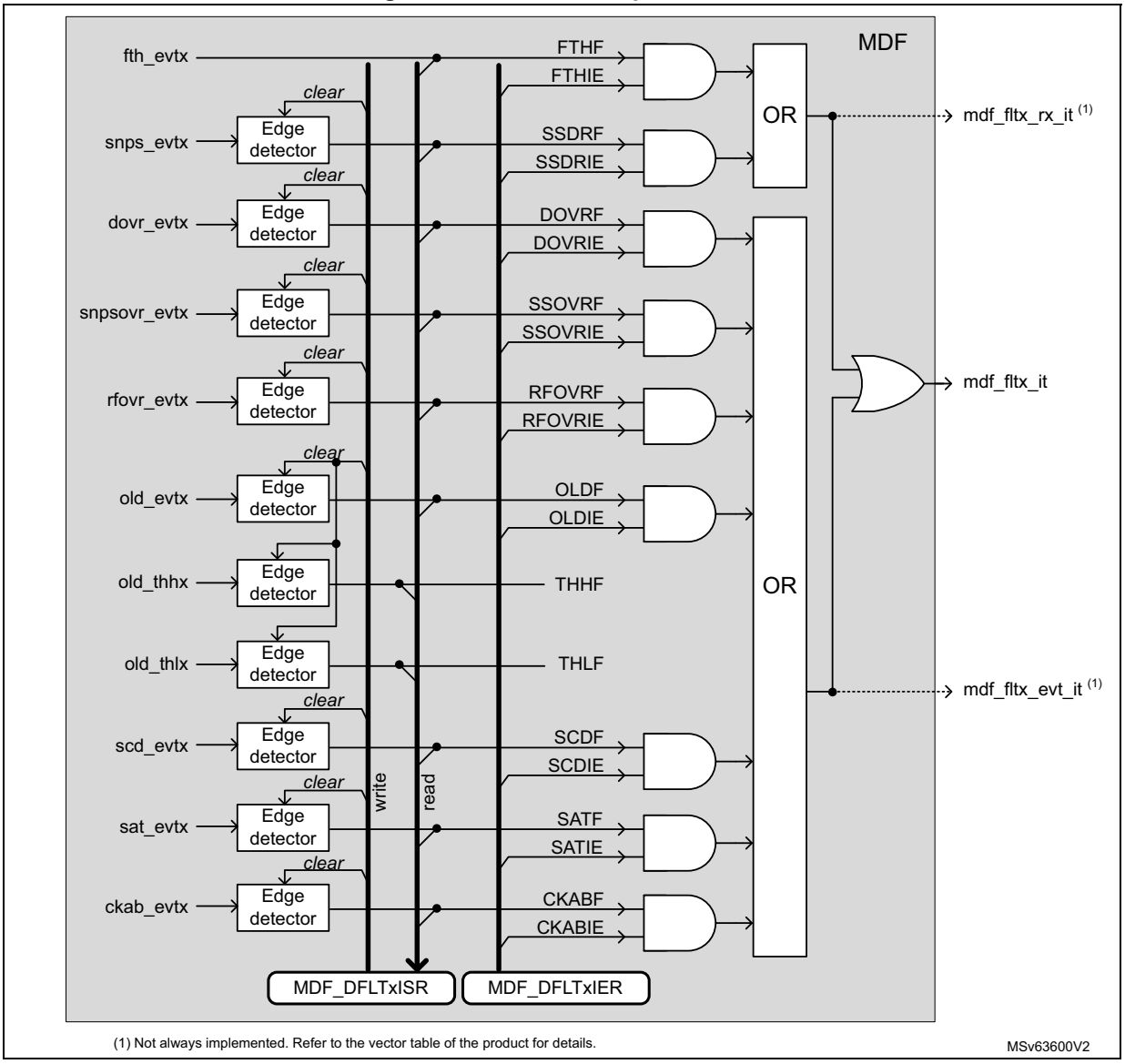
To increase the CPU performance, the MDF offers the following interrupt lines per digital filter:

- receive interrupt MDF_FLTx_RX (mdf_fltx_rx_it)
- event interrupt MDF_FLTx_EVT (mdf_fltx_evt_it)
- a combination of both interrupts MDF_FLTx (mdf_fltx_it)

Note: Interrupts are not always connected to the device (see Section 39.3: MDF implementation for more details).
The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Figure 327. MDF interrupt interface



(1) Not always implemented. Refer to the vector table of the product for details.

MSv63600V2

The table below shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 354. MDF interrupt requests

Interrupt vectors		Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes[1]	Exit Standby mode
MDF_FLTx[2]	MDF_FLTx_RX[3]	RXFIFO threshold reached	FTHF	Read MDF_DFLTxDR until RXFIFO level is lower than the threshold.	Yes	Yes	No
		Snapshot data ready	SSDRF	Write SSDRF to 1.			
	MDF_FLTx_EVT[4]	Snapshot data overrun	SSOVRF	Write SSOVRF to 1.			
		RXFIFO overrun	DOVRF	Write DOVRF to 1.			
		RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
		Short-circuit detector	SCDF	Write SCDF to 1.			
		Saturation detection	SATF	Write SATF to 1.			
		Channel clock absence detection	CKABF	Write CKABF to 1.			
		Out-of-limit detector	OLDF	Write OLDF to 1.			
-	-		THHF	Write OLDF to 1.	-	-	-
-	-		THLF	Write OLDF to 1.	-	-	-

1. Refer to *Section 39.3: MDF implementation* for details.
2. MDF_FLTx vector corresponds to the assertion of mdf_fltx_it signal
3. MDF_FLTx_RX vector corresponds to the assertion of mdf_fltx_rx signal.
4. MDF_FLTx_EVT vector corresponds to the assertion of mdf_fltx_evt signal.

39.7 MDF application informations

39.7.1 MDF configuration examples for audio capture

Table 355 gives some examples of the MDF settings for the digital microphones, focusing on 16 and 48 kHz output data rate. In these examples, the following is expected:

- The INT block is bypassed (INTVAL = 0).
- The offset error compensation is disabled (OFFSET = 0).

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (mdf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 355. Examples of MDF settings for microphone capture

Configuration	mdf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	mdf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{MDF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

39.7.2 Programming examples

This example describes how to capture sound from four microphones, assuming that each microphone pair shares the same data line. The MDF_SD0 and MDF_SD2 data lines are used. The microphone clock is provided by the MDF via MDF_CCK0 pin.

Table 356. Programming sequence

Operations	Comments
Adjust the proper kernel clock frequency via the RCC block	Assuming that the RCC is programmed to provide a kernel clock (mdf_ker_ck) of 12.288 MHz
Select the proper MDF kernel clock source via the RCC block	Refer to the RCC of the product.
Enable the MDF clocks via the RCC block	Refer to the RCC of the product.
Reset the MDF via the RCC block	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select MDF_SD0, MDF_SD2 and MDF_CCK0 function.

Table 356. Programming sequence (continued)

Operations	Comments
Enable MDF processing clock: MDF_CKGCR = 0x0103 0023	PROCDIV = 1 (division by 2): mdf_proc_ck frequency is 6.144 MHz. CCKDIV = 3 (division by 4): MDF_CCK0 clock frequency is 1.536 MHz. The MDF_CCK0 pad is set in output and generates a clock so that the microphones can exit from low-power mode.
Serial interfaces configuration: MDF_SITF0CR = 0x0000 1F01 MDF_SITF2CR = 0x0000 1F01	SCKSRC = 0 to select MDF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER mode. Clock absence feature is not working in this mode. The serial interfaces are enabled.
Bitstream matrix configuration: MDF_BSMX0CR = 0x0000 0000 MDF_BSMX1CR = 0x0000 0001 MDF_BSMX2CR = 0x0000 0004 MDF_BSMX3CR = 0x0000 0005	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge. DFLT1 filter takes the bitstream of SITF0, sampled on falling edge. DFLT2 filter takes the bitstream of SITF2, sampled on rising edge. DFLT3 filter takes the bitstream of SITF2, sampled on falling edge.
Filters configuration (CIC): MDF_DFLT0CICR = 0x02C0 1750 MDF_DFLT1CICR = 0x02C0 1750 MDF_DFLT2CICR = 0x02C0 1750 MDF_DFLT3CICR = 0x02C0 1750	SCALE = 0x2C (- 12 dB) to avoid any saturation MCICD = 0x17 (decimation by 24) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): MDF_DFLT0RSFR = 0x0000 0100 MDF_DFLT1RSFR = 0x0000 0100 MDF_DFLT2RSFR = 0x0000 0100 MDF_DFLT3RSFR = 0x0000 0100	HPFC = 1: cut-off frequency of 16 kHz * 0.00125 = 20 Hz HPFBYP = 0: HPF not bypassed RSFLTD = 0: RSFLT decimates by 4 RSFLTBYP = 0: RSFLT is not bypassed
Filters configuration (INT): MDF_DFLT0INTR = 0x0000 0000 MDF_DFLT1INTR = 0x0000 0000 MDF_DFLT2INTR = 0x0000 0000 MDF_DFLT3INTR = 0x0000 0000	INTVAL = 0: INT filter not used Other parameter is not significant.
Micro delay adjust: MDF_DLY0CR = 0x0000 0005 MDF_DLY0CR = 0x0000 0012 MDF_DLY0CR = 0x0000 0023 MDF_DLY0CR = 0x0000 0000	Initial micro-delay for each microphone, values just given as example
Offset error correction: MDF_OEC0CR = 0x0000 0000 MDF_OEC1CR = 0x0000 0000 MDF_OEC2CR = 0x0000 0000 MDF_OEC3CR = 0x0000 0000	No correction. DC offset is removed by HPF.

Table 356. Programming sequence (continued)

Operations	Comments
Short circuit detection: MDF_SCD0CR = 0x0000 0000 MDF_SCD1CR = 0x0000 0000 MDF_SCD2CR = 0x0000 0000 MDF_SCD3CR = 0x0000 0000	SCD function for each filter is disabled.
Enable interrupt events: MDF_DFLT0IER = 0x0000 0202 MDF_DFLT1IER = 0x0000 0202 MDF_DFLT2IER = 0x0000 0202 MDF_DFLT3IER = 0x0000 0202	Enable the interrupt event the application wants to handle. In this example, the SATIE and DOVRIE bits are set to 1 to have an interrupt if a saturation or an data overflow occurs.
Digital filter control: MDF_DFLT0CR = 0x0000 0027 MDF_DFLT1CR = 0x0000 0027 MDF_DFLT2CR = 0x0000 0027 MDF_DFLT3CR = 0x0000 0027 Wait for DFLTACTIVE = 0 for the filters	NBDIS = 0: no samples discarded TRGSRC = 0: TRGO selected as trigger source TRGSENS = 0: Trigger on rising edge ACQMOD = 2: Synchronous continuous acquisition mode DMAEN = 1: DMA interface enabled DFLTEN = 1: digital filter enabled
Clear status flags: MDF_DFLT0ISR = 0x0000 0FFF MDF_DFLT1ISR = 0x0000 0FFF MDF_DFLT2ISR = 0x0000 0FFF MDF_DFLT3ISR = 0x0000 0FFF	Clear all the status flags before running the filters.
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside MDF_DFLT0DR every time a DMA request is generated. Note that when the MDF is in interleaved acquisition mode, data of filters 0, 1, 2, and 3 are read via MDF_DFLT0DR register.
Start acquisition: MDF_GCR = 0x0000 0031	ILVNB = 3: interleaved mode with DFLT0,1,2, and 3 TRGO = 1 to trigger the acquisition of all filters waiting for TRGO rising edge event

39.7.3 Connection examples

Figure 328 shows simple connection examples of the MDF to external sensors:

- Picture on the left: four digital microphones connected to the MDF

  In this connection, DMIC1 is intended to be used alone. This is why it uses a dedicated clock (MDF_CCK0). DMIC1 is also connected to a dedicated data line in case the application intends to power-down DMIC2/3/4, when only DMIC1 is used. If the application keeps DMIC2/3/4 powered, DMIC2 and DMIC1 can share the same data line, reducing the connection to four I/Os.

  If the power consumption is not an issue, when a single microphone is used, all the microphones can be connected to the same common clock (MDF_CCK0) and each

microphone pair can share the same data line. In this case, only three I/Os are required.

In this example, the data transfer to memory can be performed either using the interleaved- or the independent-transfer mode.

- Picture in the center: sensor providing several data lines with a common clock

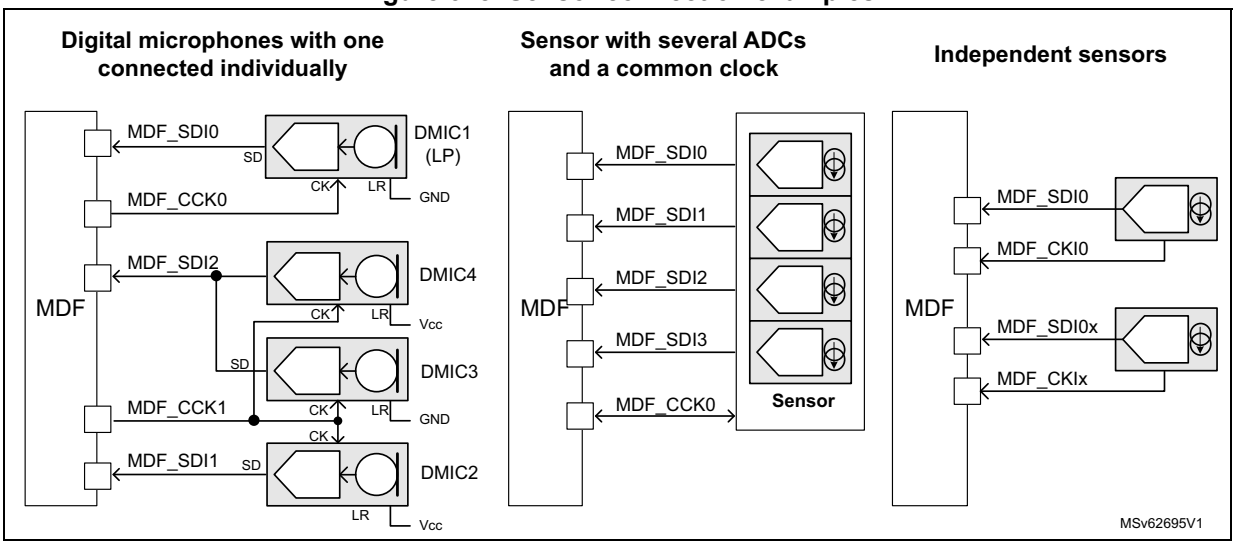
  Each data line can represent different parameters (such as current or voltage). The common clock can be provided either by the sensor or by the MDF. The data line can be shared or not by two sensors if the sensor allows it. In the figure below, the sensor does not allow the sharing of the data lines.

  In this example, the data transfer to memory can be performed either using the interleaved- or independent-transfer mode.

- Picture on the right: two independent sensors connected to the MDF

  Each of them has its dedicated clock and data lines. In this case, the data transfer to memory must use the independent-transfer mode.

Figure 328. Sensor connection examples



39.7.4 Global frequency response

Figure 329 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 40 Hz

The figure below shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 329. Global frequency response

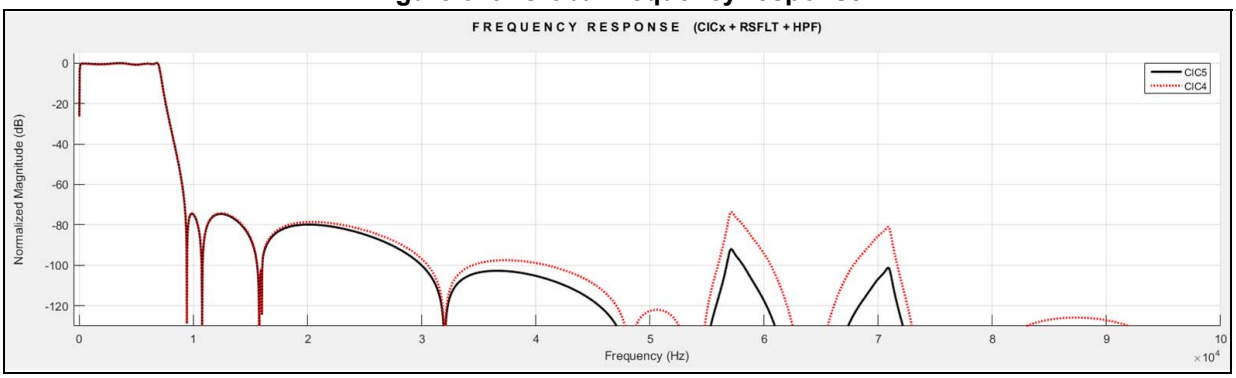


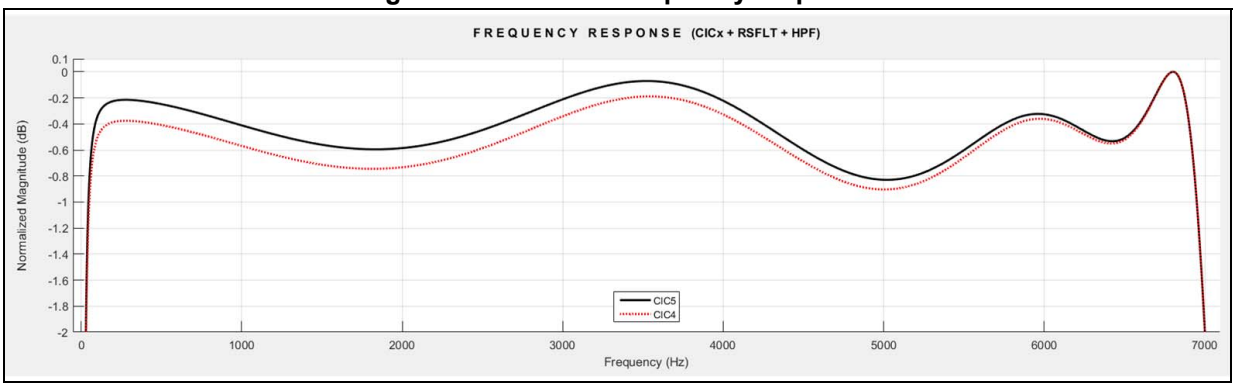
Figure 330 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 for CIC4.

The -3 dB cut-off frequency is 7061 Hz.

Figure 330. Detailed frequency response



39.7.5 Total MDF gain

This section details how to compute the signal level provided by the MDF according to the filter settings. The formula does not take into account the filters transfer function.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a - 6 dBFS level (6 dB below full scale).

For example, for the MDF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DS_{IN}$$

And the bit growth is:

$$BG_{CIC} = (N \times \log_2(D1))$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD).

DS_{IN} represents the data size (in bits) of the input signal.

Warning: DS_{CIC} is very important for CIC filters. In order to work fine, DS_{CIC} must not exceed 26 bits.

$$G_{CIC} = 2^{(BG_{CIC})} = (D1)^N$$

which gives, in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Note: The same formulas are valid for the ACIC.

Data size at SCALE output

The data size at SCALE output (including the CIC gain) is a key information as the RSFLT starts to have some saturations if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted. The resulting data size is given by:

$$DS_{SCALE} = N \times \log_2(D1) + \log_2\left(10^{\frac{GDB_{SCALE}}{20}}\right) + DS_{IN}$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits and GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.5 dB must be considered. Typically, the RSFLT increases the bit size by BG_{RSFLT}:

$$BG_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 = 1.6\ bits$$

INT gain

The INT block can also introduce a gain if the rescaling value is different from the integration value.

$$G_{INT} = \frac{IVAL}{IDIV}$$

and:

$$GdB_{INT} = 20 \times \log_{10}\left(\frac{IVAL}{IDIV}\right)$$

The bit growth of the INT is then given by the following formula:

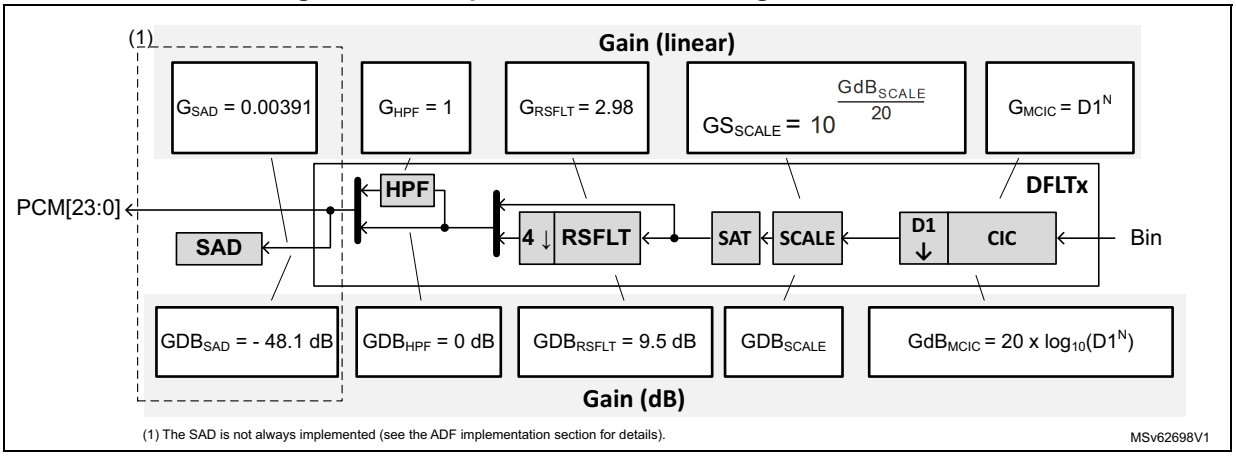
$$BG_{INT} = \log_2\left(\frac{IVAL}{IDIV}\right)$$

IVAL represents the integration value selected by INTVAL[6:0], and IDIV represents the integrator output division selected by INTDIV[1:0].

Note: The HPF filter has a gain of 0 dB.

The figure below shows a simplified view of the filter path, and gives for each significant component the expression of the bit growth and the gain.

Figure 331. Simplified DFLT view with gain information



The table below summarizes the final data size for different filter configurations.

Table 357. Output signal levels

Filter configurations	Final signal level (dBFS)	Final signal size (bits)
CIC + RSFLT (+ HPF)	$SdB_{OUT} = 20 \times \log_{10}\left(\frac{2^{DS_{OUT}}}{2^{24}}\right)$	$DS_{OUT} = DS_{SCALE} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC + RSFLT (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT}$ DS_{SCALE} must be lower than 24 bits.

Example using the main filter chain

If the MDF filter is programmed as follows:

- The input signal is coming from a serial interface (DS_{IN} = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to -12 dB.
- RSFLT is enabled and the decimation by four is enabled.
- HPF is enabled.
- INT is disabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a signal 22 bits max to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$DS_{SCALE} = DS_{CIC} + \log_2\left(10^{\frac{-12}{20}}\right) = 23.92 - 1.99 = 21.93 \text{ bits}$$

At the end, the final signal amplitude is:

$$DS_{OUT} = DS_{SCALE} + 1.6 \text{ bits} = (21.93 + 1.6) = 23.52 \text{ bits}$$

or

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.52}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

The RSFLT ripple a margin of about ± 0.41 dB (in this configuration) must be considered.

Example using the OLD filter chain

In the following example, the application wants to trigger an OLD event when the voltage coming from a shunt resistor reaches ± 200 mV.

Hypothesis:

- The ΣΔ sensor is specified to support a full-scale signal of ± 350 mV.
- The ACIC decimation ratio (D2) is fixed to 32.
- The ACIC order (N) is fixed to 3.

The ΣΔ sensor provides a full-scale digital signal (amplitude of one bit) for a signal higher or equal to ± 350 mV. If the input signal is equal 200 mV, then the digital signal amplitude provided by the sensor is 200 / 350 = 0.571.

The gain of the ACIC filter is:

$G_{CIC} = (D2)^N = 32^3 = 32768$

A signal having an amplitude of 200 mV at sensor input has then an amplitude of about 32768 * 0.571 = 18725 LSB at ACIC output.

OLDTHH must be set to 18725 and OLDTHL must be set to -18725.

39.8 MDF registers

All the MDF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

The registers are described for the MDF instance having the biggest number of filters. For registers related to filters, the number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

39.8.1 MDF global control register (MDF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is used for controls common to all digital filters.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILVNB[3:0]				Res.	Res.	Res.	TRGO
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **ILVNB[3:0]**: Interleaved number

This bitfield is set and reset by software. it enables or disables the interleaved-transfer mode and defines how many digital filters work in this mode.
This bitfield cannot be changed when DFLTEN = 1 in MDF_DFLT0CR.
0000: Interleaved-transfer mode disabled
0001: Data from DFLT0 and DFLT1 are interleaved.
0010: Data from DFLT0, DFLT1 and DFLT2 are interleaved.
...
1111: Data from DFLT0 to DFLT15 are interleaved.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also able to synchronize several MDF together by controlling the mdf_trgo signal.
0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.
1: Write 1 generates a positive pulse on mdf_trgo signal and triggers the acquisition on the enabled filters having ACQMOD[2:0] = 0x1 and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

39.8.2 MDF clock generator control register (MDF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The mdf_proc_ck clock must be enabled before enabling other MDF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. This flag must be used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (refer to *Section 39.4.15* for details). The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This bitfield is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{mdf_itf_ck} = \frac{F_{mdf_ker_ck}}{(PROCDIV + 1)}$$

This bitfield must not be changed if one of the filters is enabled (DFTEN = 1).

0: mdf_ker_ck provided to the SITF

1: mdf_ker_ck/2 provided to the SITF

2: mdf_ker_ck/3 provided to the SITF

...

127: mdf_ker_ck/128 provided to the SITF

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the MDF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the MDF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the MDF_CCK is given by the following formula:

$$F_{MDF_CCK} = \frac{F_{mdf_ker_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This bitfield must not be changed if one of the filters is enabled (DFTEN = 1).

0000: The MDF_CCK clock is mdf_proc_ck.

0001: The MDF_CCK clock is mdf_proc_ck / 2.

0010: The MDF_CCK clock is mdf_proc_ck / 3.

...

1111: The MDF_CCK clock is mdf_proc_ck / 16.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software.It is used to select which external signals trigger for the corresponding filter. This bitfield is not significant if CKGMOD = 0.

000x: TRGO selected

0010: mdf_trg[0] selected

0011: mdf_trg[1] selected

...

1111: mdf_trg[13] selected

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details.)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge. This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: MDF_CCK1 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK1 pin.

0: MDF_CCK1 pin direction is in input.

1: MDF_CCK1 pin direction is in output.

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 5 **CCK0DIR**: MDF_CCK0 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK0.

0: MDF_CCK0 pin direction is in input.

1: MDF_CCK0 pin direction is in output.

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if one of the filters is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: MDF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the MDF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK1 pad

Bit 1 **CCK0EN**: MDF_CCK0 clock enable

This bit is set and reset by software.It is used to control the generation of the bitstream clock on the MDF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK0 pad

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

39.8.3 MDF serial interface control register x (MDF_SITFxCR)

Address offset: 0x080 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 1F00

This register is used to control the serial interfaces (SITFx). The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD[1:0]		Res.	SCKSRC[1:0]		SITFEN
			rw	rw	rw	rw	rw			rw	rw		rw	rw	rw

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when the SITFACTIVE is set to 0 (refer to *Section 39.4.15* for details).

The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The serial interface is not active and can be configured if needed.

1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (refer to *Manchester mode* for details on computation).

In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This bitfield is set and cleared by software. it is used to define the serial interface type.

00: LF_MASTER SPI mode

01: Normal SPI mode

10: Manchester mode: rising edge = logic 0, falling edge = logic 1

11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This bitfield is set and cleared by software. it is used to select the clock source of the serial interface.

00: Serial clock source is MDF_CCK0.

01: Serial clock source is MDF_CCK1.

1x: Serial clock source is MDF_CKIx (not allowed in LF_MASTER SPI mode).

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is et and cleared by software. It is used to enable/disable the serial interface.

0: Serial interface disabled

1: Serial interface enabled

39.8.4 MDF bitstream matrix control register x (MDF_BSMXxCR)

Address offset: 0x084 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to the corresponding digital filter and to the SCD. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
											rw	rw	rw	rw	rw

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when the BSMXACTIVE is set to 0. Th is BSMXACTIVE flag is a logical OR between OLDACTIVE, DFLTACTIVE, and SCDACTIVE flags. Both of them must be set to 0 in order update BSSEL[4:0] bitfield.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream Selection

This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLTx and SCDx. The size of this bitfield depends on the number of DFLTx instantiated. If this bitfield selects a not instantiated input, the MDF selects the valid stream bsx_f having the higher index number.

00000: bs0_r provided to DFLTx and SCDx

00001: bs0_f provided to DFLTx and SCDx

00010: bs1_r provided to DFLTx and SCDx (if instantiated)

00011: bs1_f provided to DFLTx and SCDx (if instantiated)

...

11110: bs15_r provided to DFLTx and SCDx (if instantiated)

11111: bs15_f provided to DFLTx and SCDx (if instantiated)

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

39.8.5 MDF digital filter control register x (MDF_DFLTxCR)

Address offset: 0x088 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT
r	r			rw	rw	rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: Digital filter active flag

This bit is set and cleared by hardware. It indicates if the digital filter is active: can be running or waiting for events.

0: Digital filter not active (can be re-enabled again, via DFLTEN, if needed)

1: Digital filter active

Bit 30 **DFLTRUN**: Digital filter run status flag

This bit is set and cleared by hardware. It indicates if the digital filter is running or not.

0: Digital filter not running and ready to accept a new trigger event

1: Digital filter running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This bitfield is set and cleared by software. it is used to define the number of samples to be discarded every time the DFLTx is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **SNPSFMT**: Snapshot data format

This bit is set and cleared by software. It is used to select the data format for the snapshot mode.

0: Integrator counter (INT_CNT) not inserted into MDF_SNPSxDR, leaving a data resolution of 23 bits

1: Integrator counter (INT_CNT) inserted at position [15:9] of MDF_SNPSxDR, leaving a data resolution of 16 bits.

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter.

0000: TRGO selected

0001: OLDx event selected

0010: mdf_trg[0] selected

...

1111: mdf_trg[13] selected

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: Digital filter trigger sensitivity selection

this bit is set and cleared by software. It is used to select the trigger sensitivity of the external signals

0: A rising edge event triggers the acquisition.

1: A falling edge even triggers the acquisition.

Note: When the trigger source is TRGO or OLDx event, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge, when OLDx event is selected, the sensitivity is forced to rising edge.
This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: Digital filter trigger mode

This bitfield is set and cleared by software. It is used to select the filter trigger mode.

000: Asynchronous continuous acquisition mode

001: Asynchronous single-shot acquisition mode

010: Synchronous continuous acquisition mode

011: Synchronous, single-shot acquisition mode

100: Window continuous acquisition mode

101: Synchronous snapshot mode

Others: same a 000

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold. This bit is not significant for RXFIFOs working in interleaved-transfer mode (see *Interleaved-transfer mode* for details).

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 0 **DFLTEN**: Digital filter enable

This bit is set and cleared by software. It is used to control the start of acquisition of the corresponding digital filter path. This bit behavior depends on ACQMOD[2:0]and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x or 100.

39.8.6 MDF digital filter configuration register x (MDF_DFLTxCICR)

Address offset: 0x08C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to control the main CIC filter.The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD 8
						rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCICD[7:0]								Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (refer to *Table 348* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

000000: 0 dB

000001: + 3.5 dB

000010: + 6 dB or shift left by 1 bit

...

011000: + 72 dB or shift left by 12 bits

100000: - 48.2 dB or shift right by 8 bits (default value)

100001: - 44.6 dB

100010: - 42.1 dB or shift right by 7 bits

100011: - 38.6 dB

...

101110: - 6 dB or shift right by 1 bit

101111: - 2.5 dB

Others: Reserved

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection

This bitfield is set and cleared by software. It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3..
3: Decimation ratio is 4
...
511: Decimation ratio is 512.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC mode

This bitfield is set and cleared by software.It is used to select the configuration and the order of the MCIC. When CICMOD[2:0] = 0xx, the CIC is split into two filters: the main CIC (MCIC) and the auxiliary CIC (ACIC, used for the out-off limit detector).

000: CIC split in two filters and MCIC configured in FastSinc filter
001: CIC split in two filters and MCIC configured in $Sinc^1$ filter
010: CIC split in two filters and MCIC configured in $Sinc^2$ filter
011: CIC split in two filters and MCIC configured in $Sinc^3$ filter
100: CIC configured in single $Sinc^4$ filter
others: CIC configured in single $Sinc^5$ filter

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter

This bitfield is set and cleared by software.
0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

39.8.7 MDF reshape filter configuration register x (MDF_DFLTxRSFR)

Address offset: 0x090 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filters. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
						rw	rw	rw			rw				rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency

This bitfield is set and cleared by software. It is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.

00: Cut-off frequency = 0.000625 x F_{PCM}

01: Cut-off frequency = 0.00125 x F_{PCM}

10: Cut-off frequency = 0.00250 x F_{PCM}

11: Cut-off frequency = 0.00950 x F_{PCM}

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bit 7 **HPFBYP**: High-pass filter bypass

This bit is set and cleared by software. It is used to bypass the high-pass filter.

0: HPF not bypassed (default value)

1: HPF bypassed

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio

This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter.

0: Decimation ratio is 4 (default value).

1: Decimation ratio is 1.

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass

This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.

0: Reshape filter not bypassed (default value)

1: Reshape filter bypassed

Note: This bit can be write-protected (refer to Section 39.4.15 for details).

39.8.8 MDF integrator configuration register x (MDF_DFLTxINTR)

Address offset: 0x094 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to the integrator (INT) settings. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INTVAL[6:0]							Res.	Res.	INTDIV[1:0]	
					rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **INTVAL[6:0]**: Integration value selection

This bitfield is set and cleared by software. It is used to select the integration value.

0: The integration value is 1, meaning bypass mode (default after reset).

1: The integration value is 2.

2: The integration value is 3.

...

127: The integration value is 128.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **INTDIV[1:0]**: Integrator output division

This bitfield is set and cleared by software. It is used to rescale the signal at the integrator output in order keep the data width lower than 24 bits.

00: The integrator data outputs are divided by 128 (default value).

01: The integrator data outputs are divided by 32.

10: The integrator data outputs are divided by 4.

11: The integrator data outputs are not divided.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

39.8.9 MDF out-of limit detector control register x (MDF_OLDxCR)

Address offset: 0x098 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to configure the OLDx.The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OLDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACICD[4:0]					Res.
r										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ACICN[1:0]		Res.	Res.	Res.	Res.	BKOLD[3:0]				Res.	Res.	THINB	OLDEN
		rw	rw					rw	rw	rw	rw			rw	rw

Bit 31 **OLDACTIVE**: OLDx active flag

This bit is set and cleared by hardware. It is used to check if the OLDx is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the OLDACTIVE is set to 0 (refer to *Section 39.4.15* for details).

The delay between a transition on OLDEN and a transition on OLDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: OLDx not active and can be configured if needed

1: OLDx active and protected fields cannot be configured

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **ACICD[4:0]**: OLDx CIC decimation ratio selection
This bitfield is set and cleared by software. It is used to select the decimation ratio of the ACIC. It is only taken into account by the MDF when CICMOD[2:0] = 0xx. The decimation ratio is given by (ACICD + 1).
0: Decimation ratio is 1.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
31: Decimation ratio is 32.
Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:12 **ACICN[1:0]**: OLDx CIC order selection
This bitfield is set and cleared by software. It is used to select the ACIC type and order. It is only taken into account by the MDF when CICMOD[2:0] = 0xx.
00: FastSinc filter type
01: $Sinc^1$ filter type
10: $Sinc^2$ filter type
11: $Sinc^3$ filter type
Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKOLD[3:0]**: Break signal assignment for out-of limit detector
This bitfield is set and cleared by software.
BKOLD[i] = 0: Break signal (mdf_break[i]) not assigned to threshold event
BKOLD[i] = 1: Break signal (mdf_break[i]) assigned to threshold event
Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **THINB**: Threshold In band
This bit is set and cleared by software.
0: The OLDx generates an event if the signal is lower than OLDTHL or higher than OLDTHH (default value).
1: The OLDx generates an event if the signal is lower than OLDTHH and higher than OLDTHL.
Note: This bit can be write-protected (refer to Section 39.4.15 for details).

Bit 0 **OLDEN**: OLDx enable
This bit is set and cleared by software.
0: OLDx disabled (default value)
1: OLDx enabled, including the ACIC filter working in continuous mode

39.8.10 MDF OLDx low threshold register x (MDF_OLDxTHLR)

Address offset: 0x09C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used for the adjustment of the out-off-limit low threshold. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OLDTHL[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OLDTHL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OLDTHL[25:0]**: OLD low threshold value

This bitfield is set and cleared by software. OLDTHL represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHL.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

39.8.11 MDF OLDx high threshold register x (MDF_OLDxTHHR)

Address offset: 0x0A0 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used for the adjustment of the OLDx high threshold. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OLDTHH[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OLDTHH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OLDTHH[25:0]**: OLDx high threshold value

This bitfield is set and cleared by software. OLDTHH represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHH.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

39.8.12 MDF delay control register x (MDF_DLYxCR)

Address offset: 0x0A4 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **SKPBF**: Skip busy flag

This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.

0: MDF ready to accept a new value into SKPDLY[6:0]

1: Last valid SKPDLY[6:0] still under processing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This bitfield is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and the corresponding DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

39.8.13 MDF short circuit detector control register x (MDF_SCDxCR)

Address offset: 0x0A8 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCDT[7:4]			
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCDT[3:0]				Res.	Res.	Res.	Res.	BKSCD[3:0]				Res.	Res.	Res.	SCDEN
rw	rw	rw	rw					rw	rw	rw	rw				rw

Bit 31 **SCDACTIVE**: SCDx active flag

This bit is set and cleared by hardware. It is used to check if the SCDx is effectively enabled (active) or not. The protected fields of this function can only be updated when the SCDACTIVE is set to 0 (refer to *Section 39.4.15* for details).

The delay between a transition on SCDEN and a transition on SCDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: SCDx not active and can be configured if needed

1: SCDx active and protected fields cannot be configured

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:12 **SCDT[7:0]**: SCDx threshold

This bitfield is set and cleared by software. These bits are written by software to define the threshold counter for SCDx. If this value is reached, a short-circuit detector event occurs on a given input stream.

0: 2 consecutive 1's or 0's generate an event.

1: 2 consecutive 1's or 0's generate an event.

2: 3 consecutive 1's or 0's generate an event.

...

255: 256 consecutive 1's or 0's generate an event.

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKSCD[3:0]**: Break signal assignment for short circuit detector

This bitfield is set and cleared by software.

BKSCD[i] = 0: Break signal (mdf_break[i]) not assigned to this SCD event

BKSCD[i] = 1: Break signal (mdf_break[i]) assigned to this SCD event

Note: This bitfield can be write-protected (refer to Section 39.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SCDEN**: SCDx enable

This bit is set and cleared by software.

0: SCDx disabled

1: SCDx enabled,

39.8.14 MDF DFLT0 interrupt enable register 0 (MDF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCD0 interrupt enable
This bit is set and cleared by software.
0: SCD0 interrupt disabled
1: SCD0 interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLD0 interrupt enable
This bit is set and cleared by software.
0: OLD0 event interrupt disabled
1: OLD0 event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

39.8.15 MDF DFLTx interrupt enable register x (MDF_DFLTxIER)

Address offset: 0x12C + 0x80 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register is used for allowing or not, the events to generate an interrupt. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCDx interrupt enable
This bit is set and cleared by software.
0: SCDx interrupt disabled
1: SCDx interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLDx interrupt enable
This bit is set and cleared by software.
0: OLDx event interrupt disabled
1: OLDx event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

39.8.16 MDF DFLT0 interrupt status register 0 (MDF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD0 event is detected. Write 0 has no effect.
1: Read 1 means that a SCD0 event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.
1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was lower than OLDTHH when the last OLD0 event occurred.

1: The signal was higher than OLDTHH when the last OLD0 event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLD0 event occurred.

1: The signal was lower than OLDTHL when the last OLD0 event occurred.

Bit 4 **OLDF**: OLD0 flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLD0 event is detected. Write 0 has no effect.

1: Read 1 means that an OLD0 event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

39.8.17 MDF DFLTx interrupt status register x (MDF_DFLTxISR)

Address offset: 0x130 + 0x80 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD event is detected. Write 0 has no effect.
1: Read 1 means that a SCD event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.
1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag
This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.
0: The signal was lower than OLDTHH when the last OLDx event occurred.
1: The signal was higher than OLDTHH when the last OLDx event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLDx event occurred.

1: The signal was lower than OLDTHL when the last OLDx event occurred.

Bit 4 **OLDF**: OLDx flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLDx event is detected. Write 0 has no effect.

1: Read 1 means that an OLDx event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

39.8.18 MDF offset error compensation control register x (MDF_OECxCR)

Address offset: 0x0B4 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register contains the offset compensation value. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OFFSET[25:0]**: Offset error compensation
This bitfield is set and cleared by software. If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back this bitfield informs the application on the current offset value.
This bitfield represents the value to be subtracted to the signal before going to the SCALE.

39.8.19 MDF snapshot data register x (MDF_SNPSxDR)

Address offset: 0x0EC + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter in snapshot mode. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTSDR[6:0]							MCICDC[8:0]								
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **SDR[15:0]**: Contains the 16 MSB of the last valid data processed by the digital filter.

Bits 15:9 **EXTSDR[6:0]**: Extended data size
If SNPSFMT = 0, this bitfield contains the bits 7 to 1 of the last valid data processed by the digital filter.
If SNPSFMT = 1, this bitfield contains the INT accumulator counter value when the last trigger event occurs (INT_CNT).

Bits 8:0 **MCICDC[8:0]**: Contains the MCIC decimation counter value when the last trigger event occurs (MCIC_CNT)

39.8.20 MDF digital filter data register x (MDF_DFLTxDR)

Address offset: 0x0F0 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter. The number of registers is equal to the amount of filters. Refer to *Section 39.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:8 **DR[23:0]**: Data processed by digital filter

Bits 7:0 Reserved, must be kept at reset value.

39.8.21 MDF hardware configuration register (MDF_HWCFGR)

Address offset: 0xFF0

Reset value: 0x0001 0404

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTION_REGOUT[7:0]								SAD[3:0]				FLT_CFG[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBF[7:0]								FIFO_SIZE[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **OPTION_REGOUT[7:0]:** Support of MDF_OR register
0: MDF_OR not implemented
1: MDF_OR implemented with only one control bit at position 0
2: MDF_OR implemented, with two control bits at positions 0 and 1
...
32: MDF_OR implemented with 32 control bits at positions 0 to 31
Others: Invalid values

Bits 23:20 **SAD[3:0]**: Sound activity detection
0x0: SAD not implemented
0x1: SAD implemented

Bits 19:16 **FLT_CFG[3:0]**: Digital filter configuration
0: Reduced set of features to mainly address audio use-cases
1: Full set of features, with associated blocks for motor control applications

Bits 15:8 **NBF[7:0]**: Number of digital filter paths and serial interfaces
0x1: 1 digital filter instance implemented
0x2: 2 digital filter instances implemented
...
0x6: 6 digital filter instances implemented
...
0x10: 16 digital filter instances implemented

Bits 7:0 **FIFO_SIZE[7:0]**: FIFO size for MDF
2: the FIFO depth is 2 words
4: the FIFO depth is 4 words
8: the FIFO depth is 8 words
16: the FIFO depth is 16 words
32: the FIFO depth is 32 words
64: the FIFO depth is 64 words
Others: Invalid values

39.8.22 MDF version register (MDF_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: MDF major revision

Bits 3:0 **MINREV[3:0]**: MDF minor revision

39.8.23 MDF identification register (MDF_IPIDR)

Address offset: 0xFF8

Reset value: 0x0011 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: MDF identifier value

39.8.24 MDF size identification register (MDF_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits returns the size of the memory region allocated to the MDF registers. The size of this memory region is of 4 Kbytes.

39.8.25 MDF register map

Table 358. MDF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	MDF_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILVNB [3:0]				Res.	Res.	Res.	TRGO
	Reset value																									0	0	0	0				0
0x004	MDF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV [3:0]			TRGSRC [3:0]					Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0
0x008-0x07C	Reserved																																
0x080 + 0x80 * x (x = 0 to 3)	MDF_SITFxCR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD [1:0]		Res.	SCKSRC [1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084 + 0x80 * x (x = 0 to 3)	MDF_BSMXxCR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088 + 0x80 * x (x = 0 to 3)	MDF_DFLTxCR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT	TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD [2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0				0	0	0	0	0				0		0	0	0		0	0	0
0x08C + 0x80 * x (x = 0 to 3)	MDF_DFLTxCICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD [2:0]			Res.	Res.	DATSRC [1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090 + 0x80 * x (x = 0 to 3)	MDF_ DFLTxRSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 + 0x80 * x (x = 0 to 3)	MDF_DFLTxINTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTVAL[6:0]							Res.	Res.	INTDIV [1:0]	
	Reset value																						0	0	0	0	0	0	0			0	0
0x098 + 0x80 * x (x = 0 to 3)	MDF_OLDxCR	OLDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACICD[4:0]					Res.	Res.	Res.	ACICN [1:0]		Res.	Res.	Res.	Res.	BKOLD [3:0]				Res.	Res.	THINB	OLDEN
	Reset value	0										0	0	0	0	0				0	0					0	0	0	0			0	0

Table 358. MDF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x09C + 0x80 * x (x = 0 to 3)	MDF_OLDxTHLR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHL[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x80 * x (x = 0 to 3)	MDF_OLDxTHHR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHH[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 + 0x80 * x (x = 0 to 3)	MDF_DLYxCR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8 + 0x80 * x (x = 0 to 3)	MDF_SCDxCR	SCDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCDT[7:0]								Res.	Res.	Res.	Res.	BKSCD [3:0]				Res.	Res.	Res.	SCDEN
	Reset value	0												0	0	0	0	0	0	0	0					0	0	0	0				0
0x0AC	MDF_DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x0B0	MDF_DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0B4 + 0x80 * x (x = 0 to 3)	MDF_OECxCR	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8-0x0E8	Reserved																																
0x0EC + 0x80 * x (x = 0 to 3)	MDF_SNPSxDR	SDR[15:0]																EXTSDR[6:0]							MCICDC[8:0]								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F0 + 0x80 * x (x = 0 to 3)	MDF_DFLTxDR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x12C + 0x80*(x-1) (x=1 to 3)	MDF_DFLTxIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x130 + 0x80*(x-1) (x=1 to 3)	MDF_DFLTxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x474-0xFEC	Reserved																																
0xFF0	MDF_HWCFGR	OPTION_REGOUT [7:0]								SAD[3:0]				FLT_CFG [3:0]				NBF[7:0]							FIFO_SIZE[7:0]								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 358. MDF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	MDF_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFF8	MDF_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
0xFFC	MDF_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

40 Camera subsystem

40.1 Introduction

The camera subsystem is built around a double path:

- a low-resolution path, with the DCMI and a low-frequency parallel interface. It supports sensors up to 1 Mpixels @15 fps.
- a high-resolution path, with the DCMIPP and a high-frequency parallel interface or serial CSI-2 interface, either RGB or RawBayer. It targets sensors up to 5 Mpixels @30 fps.

When plugging a camera sensor, it is recommended to use the DCMIPP that supports more features and provides higher throughput than the DCMI.

The DCMI is recommended only in two noticeable cases:

- to get a backward compatibility with former platforms that embed also the DCMI
- to input the pixels from a second camera sensor (such as low resolution sensor for AI processing)

Note: The DCMIPP inputs pixels from one sensor via the CSI-2 interface, while the DCMI gets pixels from the second sensor via the parallel interface.

The DCMI path offers the following summarized maximum features:

- Target sensor: 24 Mpixel/s 16 bpp (limited by the DMA)
- Parallel input: 80 MHz on 14-bit input capability
- Central DMA to extract and dump its pixels
- Software extraction of its pixel data

The DCMIPP path offers the following summarized maximum features:

- Target sensors: 5 Mpixel at 30 fps max
- Parallel input: 120 MHz on 16-bit input capability
- Serial input: CSI-2 @2.5 Gbps/lane on 2 data lanes
- ISP (image signal processor): demosaicing, exposure, white-balance, contrast, bad-pixel, histograms
- Application pipes: 2 pipes with crop, downsize, gamma, YUV conversion, YUV420

Some over-target use cases are possible, but with specific constraints:

- sensors with resolution above 5 Mpixels
- sensors with pixel rate above 150 Mpixel/s
- double sensors in parallel

40.2 Camera path main features

40.2.1 DCMI path main features

The DCMI path shows the following features:

- 8-, 10-, 12- or 14-bit parallel interface at 80 MHz (no I/O ReTime)
- Embedded/external line and frame synchronization

- Continuous or snapshot mode
- Crop feature
- Following data formats supported (dumped in memory, extracted by software):
  - 8-, 10-, 12-, 14-bit progressive video, either monochrome or RawBayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG (in fact, any default/byte data)

See *Section 41: Digital camera interface (DCMI)* for more details.

40.2.2 DCMIPP path main features

The DCMIPP path groups the DCMIPP pixel pipeline block, its included parallel interface, and the abutted CSI-2-Host and PHY.

The parallel interface shows the following features:

- Input rate: 120 MHz up to 16-bit capability (with I/O ReTime)
- Pixel format: RGB565, 888, YUV422, RawBayer/Mono 8/10/12/14
- Synchronization: embedded versus external line and frame synchronization

See *Section 42: Digital camera interface pixel pipeline (DCMIPP)* for more details.

The CSI-2 serial interface shows the following features:

- Standard: MIPI CSI2 v1.3
- Input rate: 2 data lanes @ 2.5 Gbps/lane
- Pixel rate: if RGB888: up to 200 Mpixel/s, if RawBayer 10: up to 333 Mpixel/s
- Pixel format: any MIPI CSI2 v1.3: RGB565, 888, YUV422, RawBayer
- Miscellaneous: interlaced video, interleaved packets, virtual channels

See *Section 43: CSI-2 Host (CSI)* for more details.

DCMIPP sizings and limitations

- Camera target: 5 Mpixels (typ. 2560x2048 or 2688x1944) before decimation
- Pipe resolution: 4094x4094 pixels max
  - Demosaicing: 2688 pixels wide max line-buffers (decimation available ahead)
  - Downsize: 2688 pixels wide max line-buffers
  - YUV420: 2688 pixels wide max line-buffers.
- Pipe pixel rate: 333 Mpixel/s peak.
- AXI FIFO: 5 Kbytes available, with a flexible allocation between all outputs.
- AXI master: AXI-64 bits @ 400 MHz, with FIFO for latencies up to 2 µs, 16 outstanding.

The DCMIPP provides three different pipes to process data:

- Pipe0: to dump data without any processing, branched before any ISP

  It is aimed at dumping metadata or statistics as-is from the sensor, or at dumping the full picture as RawBayer data, for a potential long but optimal software processing.
- Pipe1: main full-featured pipe (such as ISP, then downsize, YUV-multi-planar)

  It provides a picture ready for any application (such as display, video encode, or neural processing).

- Pipe2: ancillary semi-featured pipe

  Same ISP as pipe1, then downsize, but no YUV-multiplanar. It provides an RGB output for a display use.

DCMIPP ISP capabilities common to pipe1 and 2, are the following:

- Flow selection: parallel-vs-CSI2, CSI virtual packets
- Test pattern generator to emulate an external camera sensor
- Capture mode: continuous versus snapshot mode
- Pixel extraction: RGB565, 888, YUV422, RawBayer/Mono 8/10/12/14.
- Statistic removal: to remove unused metadata (like statistics) from received pictures
- Bad-pixel removal: automatic detection and correction of bad pixels from sensor array
- Decimation: one pixel every 1/2/4/8, used to fit the max 2688 pixels
- Exposure control: to amplify the exposure from 0.0 to 255.99x
- Statistic extraction: average R,G,B,L values, dynamic bins
- Demosaicing: to convert RawBayer to RGB, per region histograms
- White-balance: to adapt to sensor white and global Illumination
- Contrast: to enhance the contrast, via luminance

The DCMIPP application capabilities, replicated for pipe1 and 2, are listed below:

- Cropping: to select a region of interest or stabilize the picture
- Downsize: box-filtering, with any decimal ratio, up to 8x8
- Gamma conversion: predistortion with gamma = 2.2 for display purposes
- YUV color conversion: to convert to YUV to later store a video-friendly buffer (pipe1 only).
- YUV422/420: chroma subsampling to YUV422 or YUV420

Output pixel format:

- Pipe1 and 2: 888, 565, 422-1, Y8, ARGB and RGBA (coplanar only)
- Pipe1: to adds422-2, 420-2, 420-3 (multi-planar possibility)

The full features are detailed in *Section 41: Digital camera interface (DCMI)*.

40.3 Camera subsystem implementation

40.3.1 Hardware settings

The camera subsystem is implemented with the following specific settings:

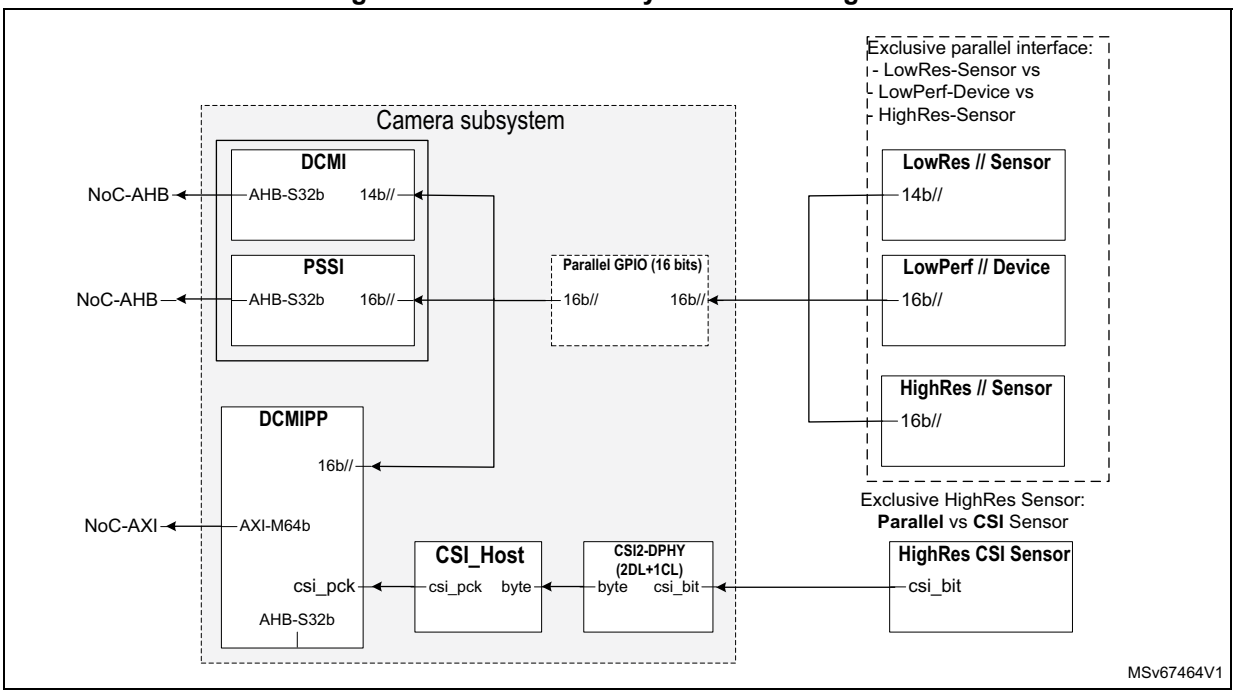
- CSI2: two data lanes (+ one clock lane) up to 2.5 Gbps/lane.
- DCMIPP FIFO: 5 Kbytes for AXI
- DCMIPP with the following subfeatures:
  - line-buffers: 2688 pixels (impacts the demosaicing, downsize, YUV420 conversion)
  - one dump pipe + two pixel pipes (pipe1 with YUV420/multi-planar only)

40.4 Camera subsystem functional description

40.4.1 Camera subsystem block diagram

The figure below shows the involved blocks in the camera subsystem.

Figure 332. Camera subsystem block diagram



Note: The PSSI is not used for camera purpose but rather for general-purpose parallel input (not described in this section).

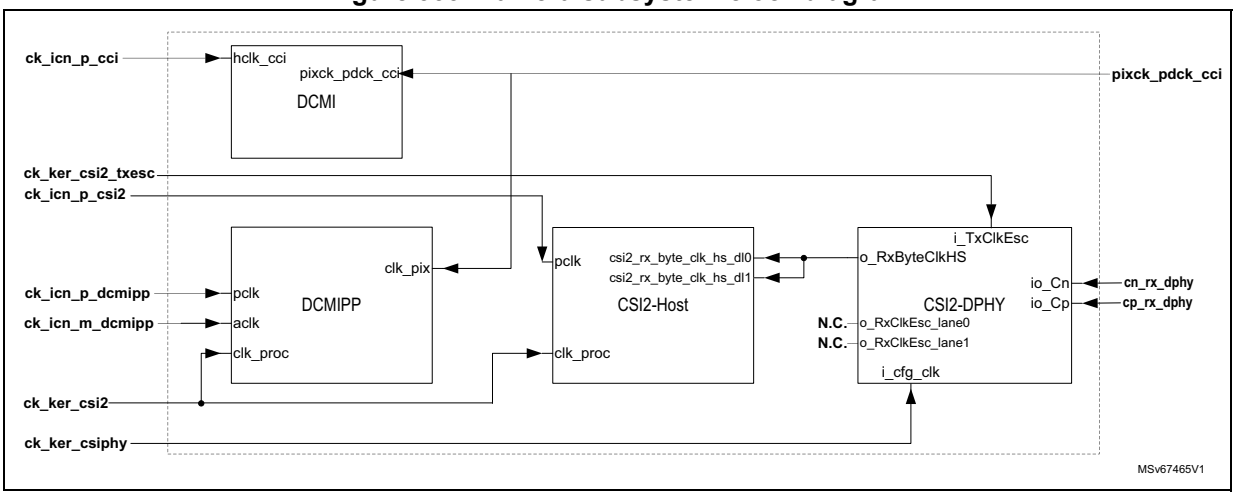
40.4.2 Camera subsystem pins and external signal interface

Details about the pins can be found in the respective block sections.

40.4.3 Camera subsystem reset and clocks

The clock diagram of the camera subsystem is shown in the figure below.

Figure 333. Camera subsystem clock diagram



The above camera subsystem has been implemented with the following clock max frequencies:

- DCMI // I/F (pixck_pdck_cci): 80 MHz (no I/O ReTime: async path from pads)
- DCMIPP // I/F (pixck_pdck_cci): 120 MHz (I/O ReTime mandatory)
- CSI2 bit clock (cn/p_rx_dphy): 1250 MHz (2.5 Gbps/lane)
- CSI2 byte clock (rxbyte_clkHS): 312.5 MHz
- CSI2 TxEsc clock (ck_ker_csi2_txesc): 20 MHz
- CSI2 Cfg clock (ck_ker_csiphy): 27 MHz
- Pixel clock (ck_ker_csi2): 333 MHz
- Register clocks (ck_icn_p_*): 150 MHz (all APB/AHB register files)
- AXI clock (ck_icn_m_dcmipp): 400 MHz

40.4.4 Camera subsystem security

The security of the camera subsystem is based on the protection of its register map, and of its master access to the system interconnect, based on default RISUP and RIMU respectively.

More information can be found in *Section 5.4: Resource isolation framework (RIF)*.

The RISUP differentiates the access right of accesses performed toward the following RIF protected peripheral ID (indexes refer to *Table 38: Indexes for non-RIF-aware peripherals (RISUP)*):

- DCMIPP: tag = DCMIPP for the configuration of pixel pipelines (index 87)
- CSI2-Host: tag = CSI2H for Host and PHY configurations (index 86)
- DCMI and PSSI: tag = DCMI for its configuration, and to retrieve pixels by DMA (index 88)

The RIMU (RIMU_AXI_DCMIPP) gives the access right to the DCMIPP AXI master:

- S and P are set using the RIF RIMC_ATTR10 register.
- CID is set either by the RISA (RISUP86 register) of directly by software.

By default, the camera subsystem is expected to work in a non-secure mode:

- DCMIPP, CSI2, DCMI, and PSSI are left unprotected, so that any OS software can configure and handle the camera subsystem.
- The AXI master of the DCMIPP inherits the CID setting from its associated register map. CID = DISP or CID = VID can be allocated to its RISUP to mention that the AXI write access of the camera subsystem contains pixels (like the display system).

For secure applications, the user can put the camera subsystem in S, P, or S-and-P mode, with the following impacts:

- As a benefit, the user is sure that the content of the dumped buffer does truly come from the camera sensor:

  The output of the DCMIPP is differentiated from other masters: only the DCMIPP can write into its buffer. This buffer content is known to come from the camera sensor, which makes impossible to fill-in (by software) a fake face to be recognized.
- The camera subsystem needs to be transfered to the secure software world (with the verification implications), or the system must make sure that the access rights of the DCMIPP cannot harm the system.

The above recommended non-protected settings are shown in *Table 359: Camera subsystem RIF peripheral ID*.

The table below lists vertically all camera subsystem blocks, and horizontally the associated RIF peripheral ID.

Table color legend for the recommended settings: gray for a permanently non-protected setting, blue for a temporarily protected versus non-protected, and pink for a permanently protected state: by default, all the settings are non-protected.

Table 359. Camera subsystem RIF peripheral ID

Subsystem	Block	Address range	Start address	RISUP - slave		
				88	87	86
				DCMI_PSSI	DCMIPP	CSI2H
Camera low resolution	DCMI	Configuration	0x0000	Y	-	-
		Pixel Rx	0x0A00	Y	-	-
		Configuration	0x0B00	Y	-	-
		Data in/out	0x0C00	Y	-	-

Table 359. Camera subsystem RIF peripheral ID (continued)

Subsystem	Block	Address range	Start address	RISUP - slave		
				88	87	86
				DCMI_PSSI	DCMIPP	CSI2H
Camera high resolution	DCMIPP	Common configuration	0x0	-	Y	-
		P1 configuration	0x0	-	Y	-
		P2 configuration	0x0	-	Y	-
		P3 configuration	0x0	-	Y	-
	CSI	Configuration control	0x0000	-	-	Y
		Configuration wrap	0x1000	-	-	Y
		CSI DPHY-Rx (no register)	-	-	-	-

40.5 Camera subsystem programmable parameters

This section describes the configuration of the various main use cases supported by the camera subsystem. Its purpose is to provide the settings that are global to the camera subsystem, and not provided by the local relevant blocks.

The use cases supported by default are the following:

- low-resolution parallel camera, using the DCMI path (see *Section 40.5.1*)
- high-resolution parallel camera, using the DCMIPP path (see *Section 40.5.2*)
- high-resolution CSI2 camera, using the DCMIPP path (see *Section 40.5.3*)

Some over-target use cases are possible, but with specific constraints:

- sensors over target-resolution (above 5 Mpixels)

  The limiting factors are
  - the clock (related to max pixel rate)
  - AXI bandwidth (related to the pixel rate and pixel format of the active pipes)
  - line buffers of the demosaicing, downsize, and YUV420 chroma decimation (when active)

  To cope with these limitations, the decimation can be used, ahead of the pipes, to reduce the width and/or height of the frame, and/or deactivate the limiting operators, the demosaicing, downsize, YUV420.
- sensors over target pixel rate (above 150 Mpixel/s)
- double sensors like 3D camera (with their left and right sensors)

40.5.1 Low-resolution parallel camera and DCMI

This use case is recommended only for backward compatibility, or to grab pictures simultaneously from two sensors (a low and a high resolution).

The steps to setup this path are the following:

- Clock and reset
  - pixck_pdck_cci is provided by the sensor (via the PCB on the input I/O, 80 MHz max).
  - The RCC provides ck_icn_p_cci, that must be ≥ 2.5x pixck_pdck_cci due to the DCMI. The default clock is 150 MHz, which allows the input clock to be at 80 MHz.
  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- DMA
  - The HPDMA retrieves pixels from the DCMI, that signals to the DMA that its FIFO is full and must be read-out.
  - *Section 25* provides the needed details.

40.5.2 High-resolution parallel camera and DCMIPP

This is the default use case to grab a picture from a parallel camera (up to the target 5 Mpixels @30fps). Any RGB or RawBayer picture can be grabbed.

The steps to setup this path are the following:

- Clock and reset:
  - pixck_pdck_cci is provided by the sensor (via the PCB on the input I/O, 120 MHz max). Each pixel needs 1 to 3 cycles to get in.
  - ck_icn_p_dcmipp is provided by the RCC for the DCMIPP configuration (default is 150 MHz).
  - ck_ker_csi2 is provided by the RCC for the pixel pipelines of the DCMIPP (default is 150 MHz).
  - ck_icn_m_dcmipp is provided by the RCC for the DCMIPP AXI (default is 400 MHz).
  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- DCMIPP configuration:
  - Select the parallel interface as input interface.
  - Setup the DCMIPP (parallel interface, ISP, application path, AXI-plug) as described in *Section 42: Digital camera interface pixel pipeline (DCMIPP)*.

40.5.3 High-resolution CSI2 camera and DCMIPP

This is the default use-case to grab a picture from a CSI2 camera (up to the target 5 Mpixels @30 fps). Any RGB or RawBayer picture can be grabbed.

The steps to setup this path are the following:

- Clock and reset:
  - cn/p_rx_dphy are provided by the sensor (via the PCB on the input I/O, 1.25 GHz/2.5 Gbit/s max).
  - RxByteClkHS is automatically divided (1/8) from cn/p_rx_dphy (312.5 MHz max).
  - ck_icn_p_csi2 is provided by the RCC for the CSI2-Host configuration (default 150 MHz).
  - ck_icn_p_dcmipp is provided by the RCC for the DCMIPP configuration (default 150 MHz).

  - ck_ker_csiphy is provided by the RCC for CSI2-Host in low-power (default 20 MHz).
  - ck_ker_csi2_txesc is provided by the RCC for the TxEscape mode of CSI2 (default 27 MHz).
  - ck_ker_csi2 is provided by the RCC for the pixel pipelines of the DCMIPP (default 150 MHz).
  - ck_icn_m_dcmipp is provided by the RCC for the DCMIPP AXI (default 400 MHz).
  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- CSI2-Host configuration:
  - Setup the CSI2-Host (such as packet type).
- DCMIPP configuration:
  - Select the CSI2 interface as input interface.
  - Setup the DCMIPP (CSI2, ISP, application path, AXI-plug) as described in *Section 42: Digital camera interface pixel pipeline (DCMIPP)*.

40.5.4 Sensors over target resolution

The sensors over the target resolution (5 Mpixels) are supported with the following restrictions:

- CSI2 interface must be used (parallel interface quickly limited with its 120 MHz performances).
- The sensor must be able to output its bit rate onto the available two data lanes of the CSI2 link. It is recommended to use RawBayer (less bit/pixel than RGB).
- Width and height must both remain below 4094 (up to 16 Mpixels can be reached).
- Decimation is located ahead of most processing, and can be used to decrease the grabbed resolution.
- Demosaicing is limited with a max width of 2688: if demosaicing needs to be activated (RawBayer input, RGB output), then the decimation must be used to fall under that 2688 pixel limit. There is no height limit.
- Downsize and YUV420 are limited (similarly as demosaicing) to 2688, and require a decimation ahead if they need to be activated. The horizontal downsize occurs before the vertical one: the first can be used to fall below the 2688 pixels limit, before being processed by the second.
- If all demosaicing, downsize, and YUV420 are disabled. Pictures up to 4094x4094 can be received and dumped by the AXI.

After taking into account these restrictions, the configuration itself is the same as for the default high-resolution CSI2 camera, described in previous section.

40.5.5 Sensors over target pixel rate

The sensors over the target pixel rate (150 Mpixel/s average, ~200 Mpixel/s peak) are supported with the following restrictions:

- The sensor must be able to output its bit rate onto the available two data lanes of the CSI2 link.
- The peak pixel rate must remain below (with some margin) the max 333 MHz of the DCMIPP pixel clock (with a 10% margin recommended).

- DCMIPP bandwidth and FIFO allocation must be able to dump this load: it is recommended to reduce the amount of parallel dumps (best not all Pipe0, Pipe1, and Pipe2 active simultaneously). In the DCMIPP plug, more FIFO and outstanding allocation can be made in favor of the working pipe.

After taking into account these restrictions, the configuration itself is the same as for the default high-resolution CSI2 camera, described in *Section 40.5.3*.

40.5.6 Double sensors

Two sensors are simultaneously and independently supported with the following restriction:

- One sensor must be low-resolution, so that it can use the DCMI (the other sensor uses the DCMIPP).
- Two input interfaces must be available: one sensor is CSI2, and the other parallel.

The configuration of the double sensors is the same as for the default low-resolution and default high-resolution sensors, described in previous sections.

40.6 Camera subsystem interrupts

Sections for each peripheral describe in details the provided interrupts.

40.7 Camera subsystem registers

All the registers are described in sections for each peripheral.

41 Digital camera interface (DCMI)

41.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

41.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

41.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

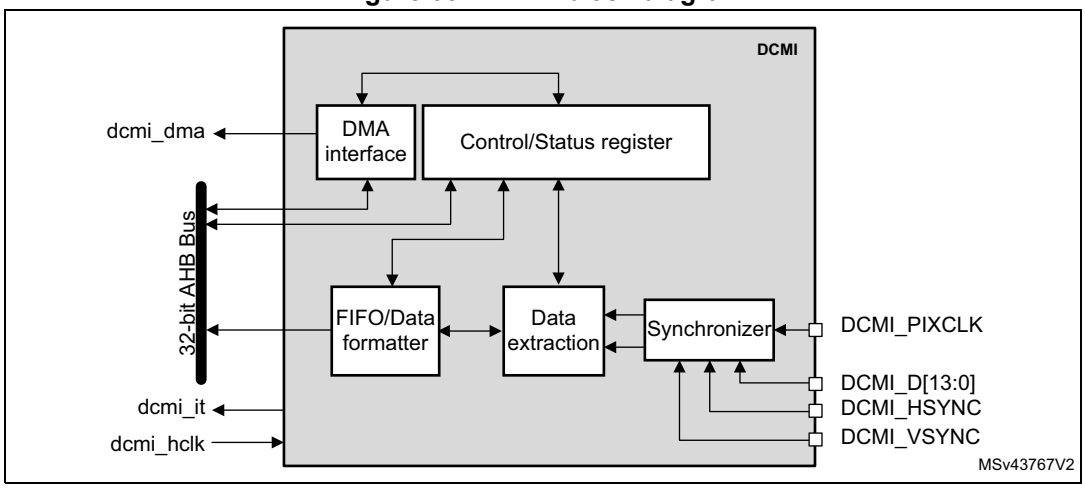
The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

41.3.1 DCMI block diagram

Figure 334 shows the DCMI block diagram.

Figure 334. DCMI block diagram



41.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

Table 360. DCMI input/output pins

Mode	Pin name	Signal type	Description
8 bits 10 bits 12 bits 14 bits	DCMI_D[7:0] DCMI_D[9:0] DCMI_D[11:0] DCMI_D[13:0]	Inputs	DCMI data
DCMI_PIXCLK		Input	Pixel clock
DCMI_HSYNC		Input	Horizontal synchronization / Data valid
DCMI_VSYNC		Input	Vertical synchronization

The following table shows DCMI internal signals.

Table 361. DCMI internal input/output signals

Internal signal name	Signal type	Description
dcmi_dma	Output	DCMI DMA request
dcmi_it	Output	DCMI interrupt request
dcmi_hclk	Input	DCMI interface clock

41.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

41.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

41.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

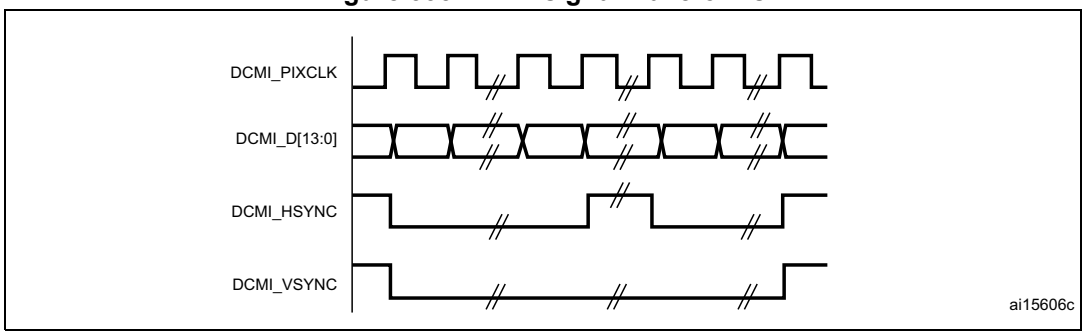
DCMI pins are shown in *Table 360*.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 335. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.

Table 362. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 363. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 364. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 365. Positioning of captured data bytes in 32-bit words (14-bit width)

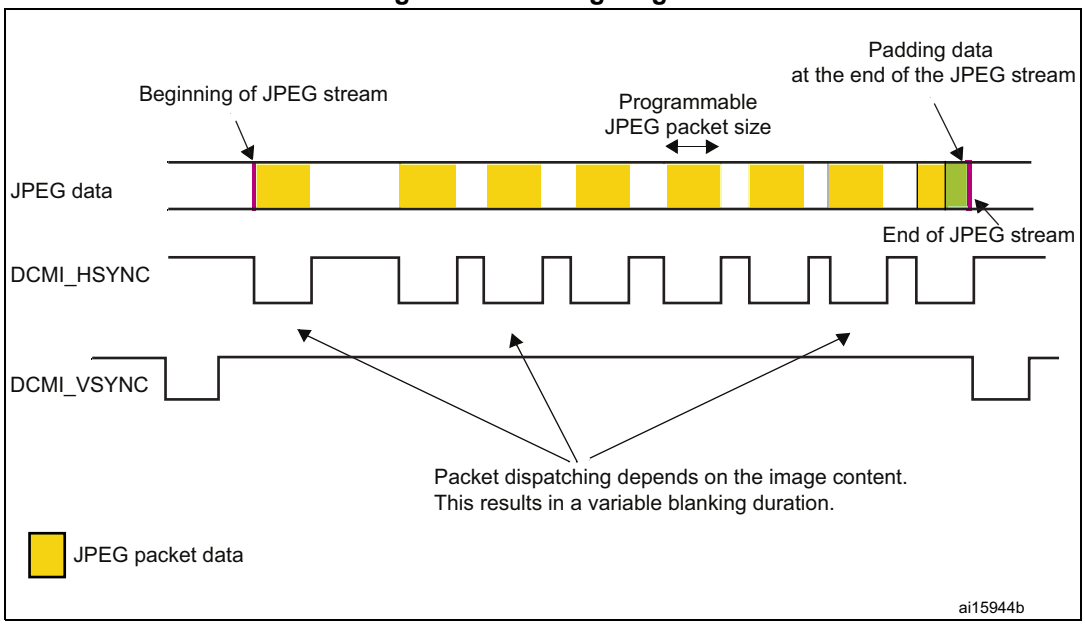
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

41.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 336* shows the corresponding timing diagram.

Figure 336. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and

capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 41.5.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS $\leq$ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and

detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

41.3.7 DCMI capture modes

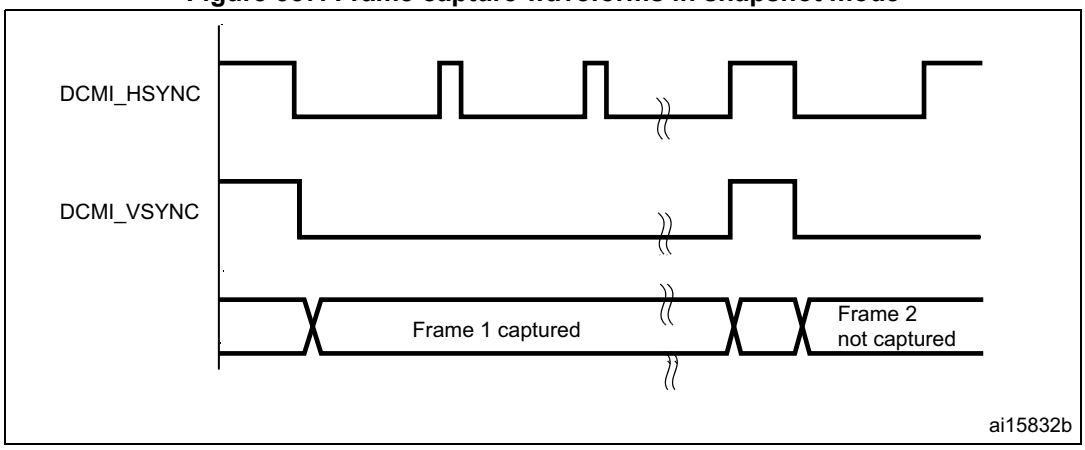
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 337. Frame capture waveforms in snapshot mode

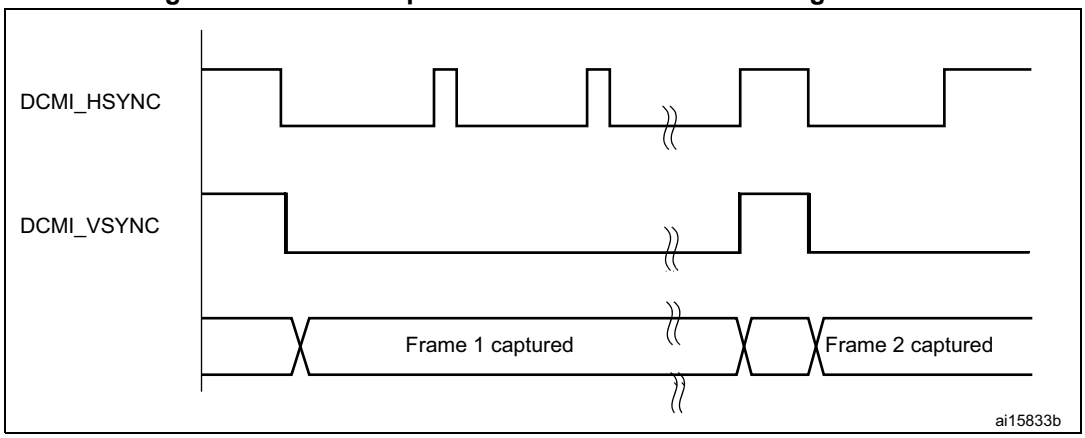


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 338. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

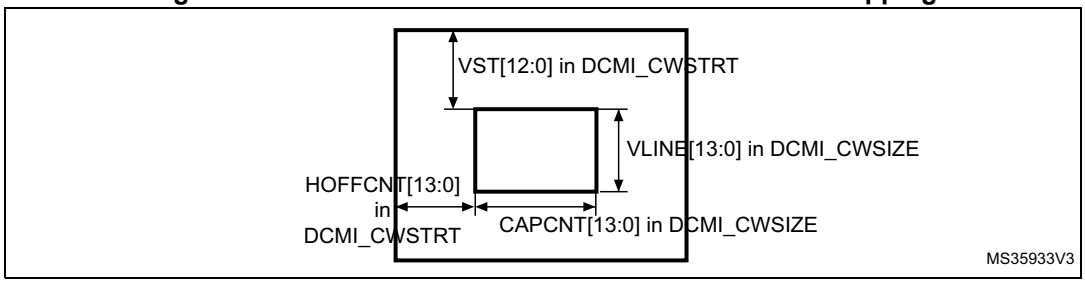
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

41.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

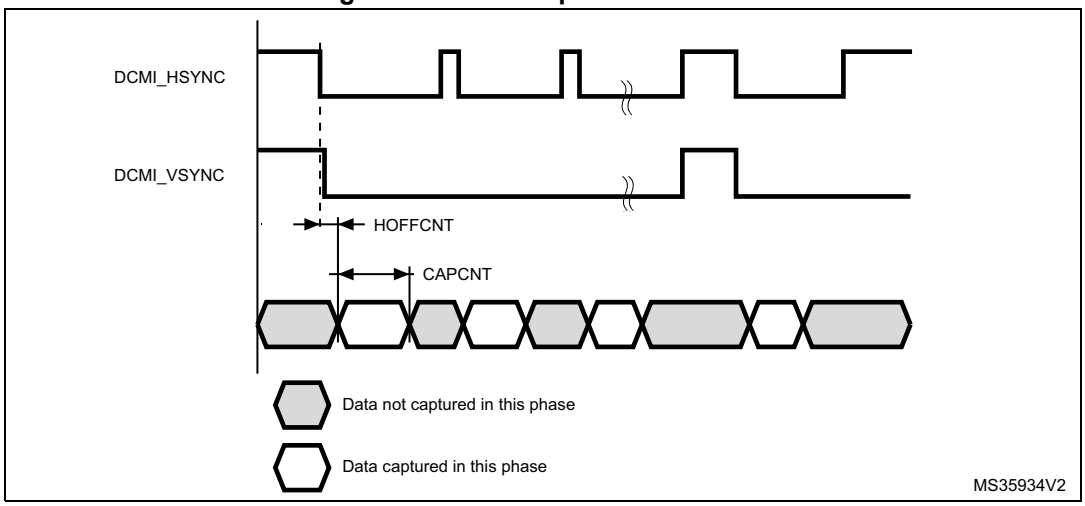
Figure 339. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 340. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

41.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

41.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

41.3.11 DCMI data format description

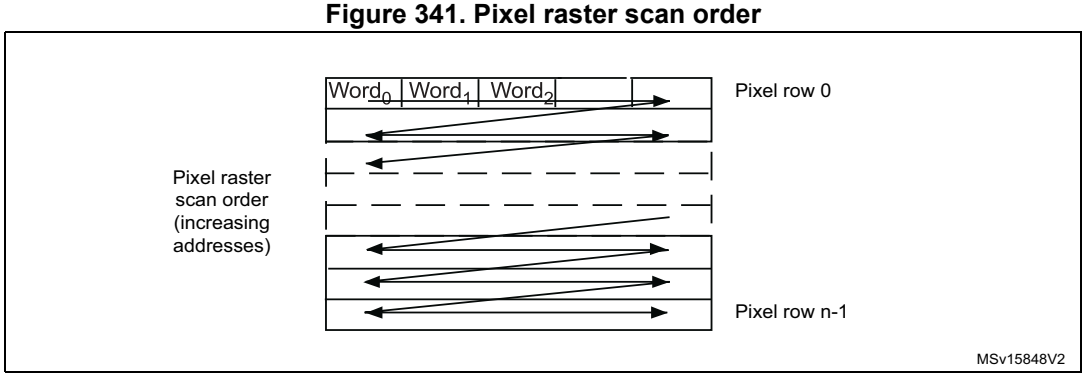
Data formats

The following types of data are supported:

- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred
- compressed data: JPEG

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 341. Pixel raster scan order



Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 366. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.

Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

Table 367. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

Table 368. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in *Table 369*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 369. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

41.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

Table 370. DCMI interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exits Sleep mode	Exists Stop and Standby modes
dcmi_it	End of line	LINE_RIS	LINE_IE	Set LINE_ISC	Yes	No
	End of frame capture	FRAME_RIS	FRAME_IE	Set FRAME_ISC	Yes	No
	Overrun of data reception	OVR_RIS	OVR_IE	Set OVR_ISC	Yes	No
	Synchronization frame	VSYNC_RIS	VSYNC_IE	Set VSYNC_ISC	Yes	No
	Detection of an error in the embedded synchronization frame detection	ERR_RIS	ERR_IE	Set ERR_ISC	Yes	No

41.5 DCMI registers

Refer to *Section 1.2 on page 167* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

41.5.1 DCMI control register (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OELS:** Odd/Even Line Select (Line Select Start)
This bit works in conjunction with the LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one being dropped.
1: Interface captures second line from the frame start, first one being dropped.

Bit 19 **LSM:** Line Select mode
0: Interface captures all received lines.
1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)
This bit works in conjunction with BSM field (BSM ≠ 00).
0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.
1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 **BSM[1:0]:** Byte Select mode
00: Interface captures all received data.
01: Interface captures every other byte from the received data.
10: Interface captures one byte out of four.
11: Interface captures two bytes out of four.
Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable
0: DCMI disabled
1: DCMI enabled
Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode
00: Interface captures 8-bit data on every pixel clock.
01: Interface captures 10-bit data on every pixel clock.
10: Interface captures 12-bit data on every pixel clock.
11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]:** Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.
00: All frames are captured.
01: Every alternate frame captured (50% bandwidth reduction)
10: One frame out of four captured (75% bandwidth reduction)
11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity
This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.
0: DCMI_VSYNC active low
1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.

0: DCMI_HSYNC active low

1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock.

0: Falling edge active

1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select

0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.

1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format

0: Uncompressed video format

1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature

0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.

1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled

1: Capture enabled

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

41.5.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO.
1: FIFO contains valid data.
0: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

41.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.

41.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received.

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

41.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt.
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt.

0: No interrupt is generated on DCMI_VSYNC transitions.

1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.

The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt.

0: No interrupt is generated on a synchronization error.

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status

This bit gives the status of the masked overflow interrupt.

0: No interrupt is generated on overrun.

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt

0: No interrupt is generated after a complete capture.

1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

41.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC**: line interrupt status clear

Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear

Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear

Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear
Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear
Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

41.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEC[7:0]								LEC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSC[7:0]								FSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEC[7:0]**: Frame end delimiter code
This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.
If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code
This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code
This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code
This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.
If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

41.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEU[7:0]								LEU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSU[7:0]								FSU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

41.5.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	VST[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HOFFCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

41.5.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	VLINE[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

41.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

41.5.12 DCMI hardware configuration register (DCMI_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_8[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **FIFO_8[3:0]:** FIFO size

41.5.13 DCMI version register (DCMI_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** IP version, major revision information

Bits 3:0 **MINREV[3:0]:** IP version, minor revision information

41.5.14 DCMI identification register (DCMI_IPIDR)

Address offset: 0x3F8

Reset value: 0x0016 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IP_ID**[31:0]: 0x0016 0011

41.5.15 DCMI size identification register (DCMI_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code

This bitfield returns the size identification code of the DCMI peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1-Kbyte address decoding)

41.5.16 DCMI register map

Table 371. DCMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]		Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 371. DCMI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	DCMI_ESUR	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C - 0x3EF	Reserved	Reserved																															
0x3F0	DCMI_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_8[3:0]			
	Reset value																													0	0	0	1
0x3F4	DCMI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
0x3F8	DCMI_IPIDR	IP_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	DCMI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

42 Digital camera interface pixel pipeline (DCMIPP)

42.1 DCMIPP introduction

The DCMIPP is the pixel pipeline section of a high-resolution camera subsystem: it gets pixels from a parallel or a CSI interface, and after some processing (such as decimation, cropping, downsize, color conversion, gamma correction, auto-exposure) dumps them to the memory.

DCMIPP supports multiple types of external sensors, among others:

- dumb sensors (without internal ISP), which output raw Bayer pixels
- smart sensors (with an internal ISP), which usually output RGB or YUV pixels
- smart sensors with internal compression, which output a bit-stream (such as JPEG, or H264)
- sensors with interlaced video, which output their odd and even fields sequentially

The DCMIPP input interface integrates a parallel interface (up to 16 bits at 120 MHz, with internal/external synchronization), and makes it possible to abut an external CSI-2 host (one to two data lanes, up to 2.5 Gbps/lane).

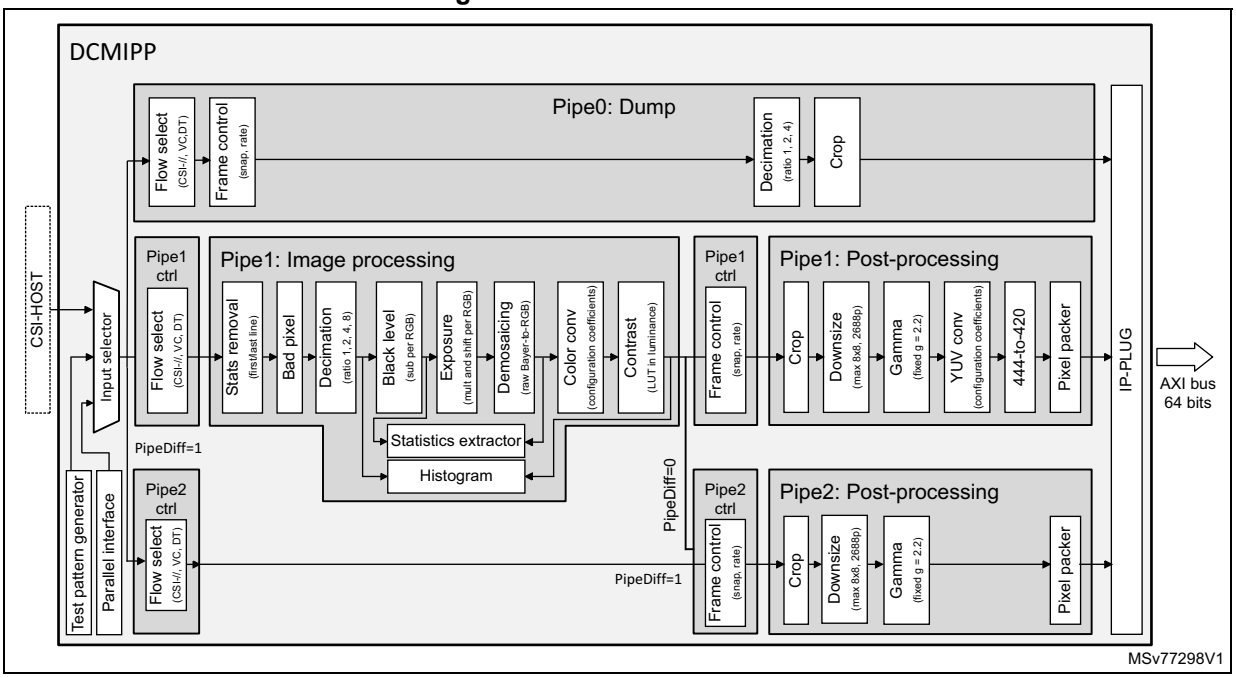
A first common part of the DCMIPP selects the input exclusively from the parallel or the CSI interfaces. Data go to dedicated pipeline(s) before they are sent to memory for further processing or display purposes.

Table 372. Available pipelines

ID	Function	Description
Pipe0	Dump pipe	Used to dump the received data as-is, without any processing (for example. without pixel extraction, no formatting), with an exception: it formats the CSI 10/12/14-bit raw Bayer components mapped unpacked into 16-bit memory words, to ease the job of a downstream application. 2D crop operation and basic decimation can be performed within this pipe.
Pipe1	Main pipe	Typically used for the main needs (such as video encode, software analysis, neural network), it allows to downsize the input, convert its color and dump to multiple pixel formats, including multi-planar ones. It can adapt the luminosity (gamma conversion) as well as auto-exposure and black level calibration based on its own embedded image processing functions, handling camera sensor without internal ISP. It can convert the raw Bayer to RGB (demosaicing). Pipe1 shares all the image processing functions with Pipe2.
Pipe2	Ancillary pipe	Typically used for ancillary needs (such as a permanent display), also allows to downsize the input and dump to several pixel formats, however restricted to coplanar ones (as specialized for display-like usage).

Figure 342 shows the DCMIPP main functions, namely the integrated parallel interface, the abutted CSI-2 host, the ISP functions in Pipe1 and shareable with Pipe2, the differentiated applicative post-processing pipelines with their crop, downsize, and pixel formatting.

Figure 342. DCMIPP overview



In the ISP processing, and therefore before the post-processing pipes (Pipe1 and Pipe2), the component dynamic is reduced to 8 bits per component. If the software intends to benefit of the wider dynamic, the input data must be dumped via Pipe0, which keeps the input pixel format (10/12/14 bpp) when storing them in memory.

Table 373. Glossary

Item	Definition
A	Alpha component, used to define transparency, opaque = 0xFF, for example ARGB.
3A (or AAA)	In ISP domain: acronym for AE-AF-AWB, algorithms automatically correcting pictures: – AE: automatic exposure control (adapts exposure or sensibility of sensor) – AF: automatic focus control (adapts the focus) – AWB: automatic white balance (adapts the color tones to have a white balance).
BPC	Bits per component (for example, RGB565 has five bits for the red component, hence 5 bpc)
BPP	Bits per pixel (for example, RGB565 has 16 bits per pixel, hence 16 bpp)
CSI	Camera serial interface, and its latest standard by MIPI®, the CSI-2 v1.3 (and v2.0)
DCMIPP_	Digital camera interface - Pixel pipeline
DT	Data type, used in the CSI-2 standard. The data type is a concept similar to the pixel format, but adds support for other data like the byte format.
Interlaced	Interlaced video: the field with odd lines is transmitted first, followed, after the last odd line, by the field with even lines. The two fields are thus consecutive but exclusive.
Interleaved	Interleaved packets, applies to CSI-2: two groups of packets transmitted by the sensor can be emitted mixed, as a packet granularity.

Table 373. Glossary (continued)

Item	Definition
ISP	Image signal processing: algorithms that retune the sensor output (through correction of pixels, demosaicing, lens-correction, light/contrast correction) to build a good quality picture.
NPU	Neural network processing engine (also known as NN, neural network).
Planar	Defines in how many sub-buffers a pixel buffer is split into: 1 (coplanar), 2 (semi-planar), 3 (full planar).
VC	Virtual channel, used in the CSI-2 standard, makes it possible to simultaneously transmit up to four parallel flows (in CSI-2 v1.3) or more (in CSI-2 v2.0).
YUV444	Pixel format, YUV color reference, all three components (Y, U, V) given per pixel.
YUV422	Pixel format, YUV color reference, chroma component (U, V) sub-sampled 1/2 in X

42.2 DCMIPP main features

- Parallel input interface:
  - Pixel rate: up to 16 bits in parallel, 100 Mpixel/s (typically 1080p30, maximum 2048x2048 on processing pipelines after decimation).
  - Pixel format: RGB565, RGB888, YUV422, raw Bayer/Mono 8/10/12/14, and ByteStream (JPEG)
- CSI-2 host input interface:
  - Pixel rate: 32 bits in parallel at 200 Mpixel/s (typically 5 MPixel x30), supports MIPI® CSI-2 v1.3
  - Pixel format: all MIPI CSI-2 v1.3, among others RGB565, RGB888, YUV422, raw Bayer (see *Table 379*)
  - Abutted CSI-2 host IP: 1-to-2 DL at up to 2.5 Gbps/lane
  - Features: Interleaved packets, four virtual channels
- Test pattern generator:
  - Color bar or square pattern, up to 16k x 16k, YUV, raw, or Mono
- Pipeline maximum resolution:
  - Dimensioned for 5 Mpixel (typically 2560 x 2048 or 2688 x 1944) before decimation
  - Maximum pipe resolution: 4094 x 4094
  - Maximum demosaicing or downsize or YUV420 width: 2688 pixels
- Pipe pixel rate: 200 Mpixel/s peak (150 Mpixel/s average + blanking)
- Maximum frame per second (fps): typical examples are defined with 30 fps, it is possible to reach higher rates (for example 120 fps) by decreasing image resolution to guarantee the maximum pixel rate constraints.
- Flow selection and frame control:
  - CSI-2 interleaved packets supported (with some restrictions)
  - Capture mode in Continuous vs. Snapshot mode
- Byte-to-pixel conversion:
  - Enables the above flexible input pixel formats to be extracted and fed to pipes
  - Non-extracted pixel formats can still be dumped via Pipe0, to allow an ad-hoc software extraction
- Statistic removal:
  - To remove unused metadata (like statistics) from received pictures
- Bad pixel removal (automatic detection and correction of bad pixels from sensor array)
- Decimation of one pixel every 1/2/4/8:
  - Decimation to fit the demosaicing (maximum 2688 pixel width)
  - Raw Bayer format support, via a decimation of components (2 by 2)
- Integrated image processing block, which makes possible to connect very low cost camera modules without any embedded ISP
  - Black level calibration
  - Statistics extraction
    - > To extract average R, G, B, L values (for auto exposure and white balance)
    - > To sort any R, G, B, L value into 12 dynamic bins (for contrast enhancement)

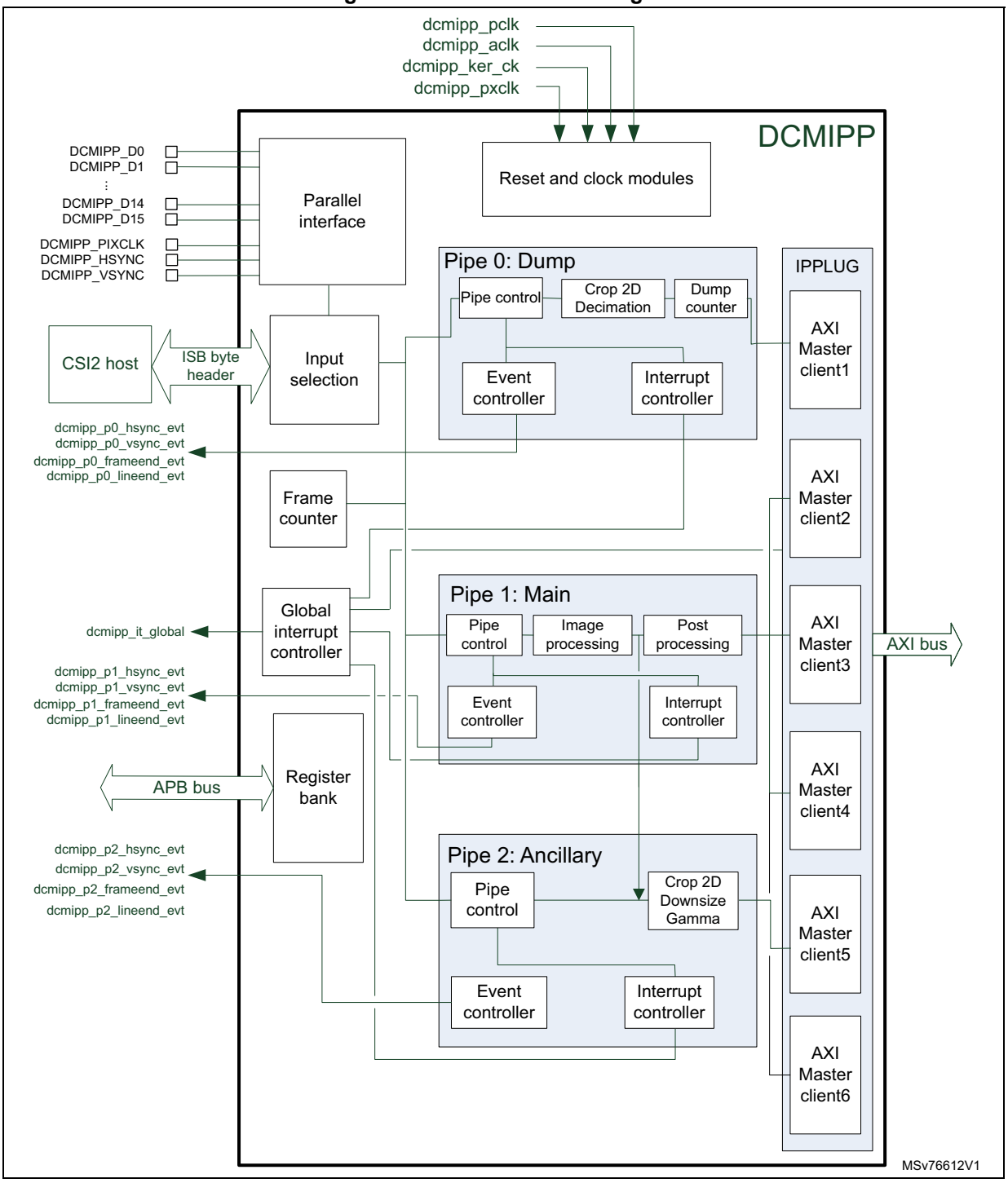
    - > Histogram extraction to extract R, Gr, Gb, B or R, G, B, L, into 4, 16, 64, or 256 bins, and from a grid (max 16x16 regions). Data are exported per frame to the system memory.
  - Exposure control (amplifies the exposure up to 256x)
    - > To smoothly adapt to dark exposures
  - Raw Bayer to RGB conversion (demosaicing)
    - > Mid quality conversion for display and software analysis purposes (high quality conversion can be done in software upon trigger, for still-image only)
    - > Algorithm with a 4 x 3 heuristic analysis matrix
  - Color conversion to adapt to the sensor and tune the illumination
    - > Configurable coefficients, to map any linear tuning
  - Contrast enhancement
    - > Non-linear luminance tuning, linearized across eight luminance segments.
- Decimation on Pipe0 (dump pipe)
  - Pure sub-sampling, ratio 1 vs. 1, 2, 4 in X and ratio 1 vs. 1, 2 in Y (no X if CSI source)
- Parallel pipelines for parallel applications:
  - Pipe0 (for data dump) for a direct dump without processing
  - Pipe1 (for main usage) with downsize, ColorConv, decimation, multi-planar
  - Pipe2 (for ancillary usage) with downsize, decimation, coplanar
- Downsize
  - Decimation, optional pre-processing, sub-sampling ratio 1 vs. 1, 2, 4, 8 in X and Y
  - Box-filtering, with any decimal ratio up to 8x8, on Pipe1 and Pipe2
- Gamma conversion for display:
  - Non-linear transform, with seven interpolated segments, with static coefficient for G = 2.2
- Color conversion 2 for RGB to YUV conversion:
  - to/from YUV: 12 coefficients (BT.601, 709, any), with optional range ext/clamp
- Output pixel format:
  - Pipe0: any data as-is, Y/Rb: 8/10/12/14 statistics, bit streams
  - Pipe2: 888, 565, 422-1, Y8, ARGB and RGBA (coplanar only)
  - Pipe1: adds 422-2, 420-2, 420-3 (multi-planar possible) to Pipe2 formats
- AXI master:
  - Output FIFOs to drain pixel/data to the memory (linked to the memory latency) from the pipe(s)
  - The unique AXI-Master interface implements one client per pipe, with semi- or multi-planar pipes using, respectively, two or three clients
- Overrun detection:
  - Notification in case of frame impacted by pixel congestion (no impact on the following frame if the congestion cause is solved)

42.3 DCMIPP functional description

42.3.1 DCMIPP block diagram

The block diagram of the DCMIPP is shown in *Figure 343*.

Figure 343. DCMIPP block diagram



42.3.2 DCMIPP pads and internal signals

Table 374. DCMIPP input/output pads

Pin name	Signal type	Description
DCMIPP_Dn (n = 0 to 15)	Input	Bit n of the parallel data bus coming from the camera sensor
DCMIPP_PIXCLK	Input	Pixel clock sent by the master (parallel camera sensor module)
DCMIPP_VSYNC	Input	VSYNC signal (vertical synchronization) coming from the camera sensor
DCMIPP_HSYNC	Input	HSYNC signal (horizontal synchronization) coming from the camera sensor

Table 375. DCMIPP input/output pins

Internal signal name	Signal type	Description
dcmipp_pclk	Input	DCMIPP APB clock (APB bus)
dcmipp_aclk	Input	DCMIPP AXI clock (AXI bus)
dcmipp_ker_ck	Input	Interface from the CSI2 host
dcmipp_it_global	Output	DCMIPP interrupts (refer to *Table 410* for the full list of interrupt sources)
dcmipp_p0_hsync_evt	Output	Pipe0 Hsync event
dcmipp_p0_vsync_evt		Pipe0 Vsync event
dcmipp_p0_frameend_evt		Pipe0 frame end event
dcmipp_p0_lineend_evt		Pipe0 line event
dcmipp_p1_hsync_evt	Output	Pipe1 Hsync event
dcmipp_p1_vsync_evt		Pipe1 Vsync event
dcmipp_p1_frameend_evt		Pipe1 frame end event
dcmipp_p1_lineend_evt		Pipe1 line event
dcmipp_p2_hsync_evt	Output	Pipe2 Hsync event
dcmipp_p2_vsync_evt		Pipe2 Vsync event
dcmipp_p2_frameend_evt		Pipe2 frame end event
dcmipp_p2_lineend_evt		Pipe2 line event

42.3.3 DCMIPP reset and clocks

Table 376. DCMIPP clocks

Domain	Clock	Maximum frequency	Comments
Parallel	dcmipp_pxclk	120 MHz	Up to 2x 120 MB/s in 16-bit mode
Pipeline	dcmipp_ker_ck	333 MHz	Up to 333 Mpixels/s (minus the blanking)
AXI bus	dcmipp_aclk	400 MHz	SoC AXI frequency
APB bus	dcmipp_pclk	200 MHz	SoC APB frequency

Table 377. DCMIPP resets

Domain	Reset type	Comments
Parallel + Pipeline	Asynchronous	Resynchronized internally on dcmipp_pxclk/dcmipp_ker_ck
AXI bus	Synchronous	Resets directly the dcmipp_aclk domain
APB bus	Synchronous	Resets directly the dcmipp_pclk domain

Clocks and pixel rate limitations

This paragraph lists the DCMIPP limitations and bottlenecks:

- The target is to sustain an average pixel rate (PixRateAvg):
  - A 5 Mpixel sensor at 30 fps requires 150 Mpixel/s (CSI-2 interface)
  - Higher resolution sensors can be supported, but with lower FPS (for example 10 Mpixels at 15 fps)
- Vertical and horizontal blanking:
  - The full system (sensor, interface, pipe, AXI) embeds limited capacity FIFOs. The idle times that the sensor inserts during its blanking periods impact all the downstream elements, including interface, pipe, and AXI. A pixel rate (PixRatePeak) higher than average must be used. The idle time due to the blanking can reach 33% of the frame duration, sometimes more (check sensor specification).
  - PixRatePeak = 1.33 * PixRateAvg = 200 Mpixel/s in our example.
- Parallel interface clock (dcmipp_pxclk):
  - On the parallel interface, pixels are received sequentially, in one, two or three cycles. The maximum implemented clock (120 MHz) constrains the peak pixel rate, depending on the ratio of cycles per pixel.
  - PixRatePeak ≤ dcmipp_pxclk / (1, 2 or 3) = 120 / (1, 2 or 3)
  - Clock frequency:
    - > dcmipp_pxclk ≤ dcmipp_ker_ck (due to resync interface between them)
    - > dcmipp_pxclk ≤ dcmipp_aclk
  - Consequences: see *Table 378*.

Table 378. Parallel interface maximum resolution (80 MHz)

Pixel		Transmission on the interface			Resolution (30 fps), including blanking	
Format	Width (bpp)	Width	Cycles/pixel	Pixel rate	Mpixel/frame	Example
Mono/raw	8	8	1	80	2.15	1080 p
	14	14	1	80	2.15	1080 p
RGB565	16	8	2	40	1.08	720 p
		16	1	80	2.15	1080 p
RGB888	24	8	3	27	0.72	qHD
		12	2	40	1.08	720 p
YUV422	16	16	1	80	2.15	1080 p
		16	2	40	1.08	720 p
YUV444	24	12	2	40	1.08	720 p
		16	2	40	1.08	720 p

  - Clock setting: dcmipp_pxclk ≥ PixRatePeak * Ratio(1,2,3)

- Pipe clock dcmipp_ker_ck (when using the CSI-2 host interface)
  - CSI-2 PHY provides data bit at the clk_bit rate, on DL parallel data lanes.
  - Rationale: internally, the CSI-2 host interface packs the serial stream on 32-bit words, resynchronized on the pipe clock (dcmipp_ker_ck).
  - Clock setting: dcmipp_ker_ck ≥ clk_bit * DL / 32 bpp = 156.25 MHz.
  - Rationale 2: The pixel pipe processes 1 pixel/clk, so that dcmipp_ker_ck must be higher than the wanted PixRatePeak.
  - Clock setting: dcmipp_ker_ck ≥ PixRatePeak.
  - For small footprint pixels (like Raw8), the DCMIPP is limited by its maximum dcmipp_ker_clk, and can be unable to handle the maximum CSI pixel rate (clk_bit * DL / 8 bpp for Raw8).
- Downsize:
  - The line buffer of the vertical downsize operator is limited to a width of 2688 pixels after the horizontal downsize operation, limiting the resolution, when the downsizing is active, to 2688 x 1944 (1944 is the usual maximum height with a width of 2688), thus 5.1 Mpixel.
  - If a camera resolution higher than 5 Mpixel is targeted, a basic decimation has to be performed to decrease the resolution to a maximum of 5 Mpixel before processing the downsizing operation.
- AXI clock:
  - The pixels must be reclocked from the processing to the AXI domain, resulting in
    > dcmipp_ker_ck ≤ dcmipp_aclk
  - The AXI bus must drain the running pipelines. The maximum footprint of pixel (or data) pipes is 32 bpp (Pipe1 and Pipe2) and 16 bpp (Pipe0). AXI is on 64 bits, with an efficiency of 8 / 9 for the typical 64-byte burst.
  - dcmipp_aclk ≥ dcmipp_ker_ck * (32 bpp + 32 bpp + 16 bpp) / (64bit / bus) * 9 / 8 (efficiency) = dcmipp_ker_ck * 1.4.

42.3.4 DCMIPP maximum resolution

The limitations related to the maximum resolution supported by the DCMIPP are:

- Maximum width and height across pipes: 4094 x 4094 pixels
  - Registers and counters are limited to 12 bits
- Maximum performance: 333 Mpixel/s on each pipe
  - Larger pixel rate can be used, but with less pipe active or lower memory latency
  - Maximum pixel rate on each pipe must remain below the pixel clock
- Maximum width on demosaicing: 2688 components wide
  - limitation related to the Line-buffer size
- Maximum width after downsize: 2688 pixels
  - limitation related to the Line-buffer size
- Maximum width for conversion to YUV420: 2688 pixels
  - limitation related to the Line-buffer size
- Max line stride: 32767 bytes
  - Registers and counters, to handle 4094 pixels x 32 bits x 2 (provision for interleaving)

42.3.5 DCMIPP minimum requirements for frame structure

DCMIPP imposes the following minimum requirements for the frame architecture:

Blanking phase

- Dump pipe
  - The minimal duration of the blanking between two frames must be an equivalent of 666 datawords of 32 bits
- Pixel pipe
  - Vertical blanking: minimum is 5 lines. If the line is less than 12 pixels, a minimum of 24 pixels equivalent for the vertical blanking parameter is requested
  - Horizontal blanking: a minimum duration of 12 pixels is required between two lines

Data image area

- Vertical image area: minimum one line (two lines if raw Bayer conversion is active)
- Horizontal image area: minimum two pixels
- The minimum total frame size is fixed to 36 pixels

42.3.6 Description of DCMIPP pixel format support

Table 379 lists the supported pixel formats in the input interfaces (parallel, CSI), as input to the pixel Pipe1 and Pipe2, and in the three output pipes.

Table 379. Supported pixel formats

Index (1)	Pixel format	BPP	Parallel input (clk/pix)(2)	CSI input	Pipe0 output (dump)	Pipe1 and Pipe2 input	Pipe1 output (main)	Pipe2 output (ancillary)
-	Bytes (CSI: long packets and user)	8	-	Y	Y	-	-	-
13(3)	Byte stream (JPEG, compressed video)	8	Y, 1	-	Y	-	-	-
14(4)	Other data	8 to 16(4)	Y, 1	-	Y	-	-	-
5	Raw6	6	(5)	Y(6)	Y(6)	-	-	-
5	Raw7	7	(5)	Y(6)	Y(6)	-	-	-
5	Raw8	8	Y, 1	Y	Y	Y	Y	Y
6	Raw10	10	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
7	Raw12	12	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
8	Raw14	14	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
9	Mono8	8	Y, 1	-	Y	Y	Y	Y
10	Mono10	10	Y, 1	-	To 16 bpc	To 8 bpc	-	-
11	Mono12	12	Y, 1	-	To 16 bpc	To 8 bpc	-	-
12	Mono14	14	Y, 1	-	To 16 bpc	To 8 bpc	-	-
2	RGB444	12	(5)	Y	(5)	To 8 bpc	-	-
2	RGB555	15	(5)	Y	(5)	To 8 bpc	-	-

Table 379. Supported pixel formats (continued)

Index (1)	Pixel format	BPP	Parallel input (clk/pix)(2)	CSI input	Pipe0 output (dump)	Pipe1 and Pipe2 input	Pipe1 output (main)	Pipe2 output (ancillary)
2	RGB565	16	Y, 1, 2	Y	Y	To 8 bpc	Y	Y
3	RGB666	18	(5)	Y	(5)	To 8 bpc	-	-
4	RGB888 / YUV444(6)	24	Y, 2, 3	Y	Y	Y	Y	Y
-	xRGB888 / xYUV444(6)	32	-	-	Y	-	Y	Y
-	RGBx888 / YUVx444(6)	32	-	-	Y	-	Y	Y
1	YUV422-1 (YUYV)	16	Y, 1, 2	Y	Y	Y	Y	Y
-	YUV422-2	16	-	Y(6)	Y(6)	-	Y	-
-	YUV420-2 (NV21/NV12)	12	-	Y(6)	Y(6)	-	Y	-
-	YUV420-3 (YV12)	12	-	Y(6)	Y(6)	-	Y	-
-	YUV422 10 bpc	20	-	Y(6)	Y(6)	-	-	-
-	YUV420 10 bpc	15	-	Y(6)	Y(6)	-	-	-
-	Any other CSI-2	Any	-	Y(6)	Y(6)	-	-	-

1. Index is used to link the input pixel format to the DCMIPP_PRCR register. The index has a meaning only for camera connected with the parallel interface
2. Pixel formats on 16 bpp (like RGB565 or YUV422-1) can be received either on an 8-bit interface (using 2 cycles), or on a 16-bit input interface (using one cycle). Pixel formats on 24 bpp (like RGB888) can be received either on an 8-bit interface (using three cycles), or on a 12-bit input interface (using two cycles).
3. In Byte stream mode, the bytes are dumped consecutively with only a 32-bit padding (if needed) at the end of the frame (or JPEG stream). There is no 32-bit padding operation inside the stream until the VSYNC deassertion (end of frame, or end of JPEG stream).
4. If the data format is different from those in this table, user can choose “others” configuration into bit field FORMAT in the DCMIPP_PRCR register, using bit-field EDM to select how many data bits must be captured in one pixel clock. It is possible to capture two 8-bit data mapped on 16-bit parallel input data in a single cycle (for instance FORMAT = Others, EDM = 100, i.e. 16-bit capture on each pixel clock). Data are 32-bit padded at the end of a line to have a complete line aligned on 32-bit width.
5. On the parallel interface input, some sensor pixel formats are supported by the DCMIPP, by selecting a wider pixel format in DCMIPP, by mapping the sensor wires onto the MSB of the DCMIPP interface. On the input missing pins, it is recommended to replicate the sensor MSB pins onto the missing input LSB pins. This helps achieving full dynamic on the extended pixel format. The work-around can be applied on the following formats (sensor output on left, mapping on DCMIPP format on right):
   - Raw6: Raw8 with 1-cycle input, processed as a 8 bpp
   - Raw7: Raw8 with 1-cycle input, processed as a 8 bpp
   - RGB444: RGB565 with 1-cycle input, processed as a 16 bpp
   - RGB555: RGB565 with 1-cycle input, processed as a 16 bpp
   - RGB666: RGB888 with 3-cycle input, processed as a 24 bpp
6. The same bit mapping is used for the input RGB888 and YUV444, the input xRGB and xYUV, and similarly for the outputs. The difference between RGB and YUV is the color reference, set in the color conversion that occurs after the input or before the output.

42.4 DCMIPP input and flow control

This section describes the functional elements of the DCMIPP. For each, it details the features, the software configuration, and provides a software configuration example (when needed).

42.4.1 DCMIPP common configuration

The DCMIPP common configuration register (DCMIPP_CMCR) enables the selection of the input interface—either parallel DCMI or serial CSI-2—from which to fetch the input pixels. It also allows swapping the R/B color components of the incoming pixels and selecting the pipe used by the frame counter.

Table 380. DCMIPP_CMCR bit function

Bit ID	Function	Comments
INSEL	Input selection	Select the input interface to fetch pixels from. The selection is common to all pipes and must not be changed while they are active. – 0: DCMI – 1: CSI-2
PSFC	Pipe selection for the frame counter	Select which pipe the frame counter maps to.
CFC	Clear frame counter	Clear the frame counter when set.
SWAPRB	Swaps R-versus-B components (and U-versus-V) of the input pixels	Operate only on pixel pipes, not on dump pipe.

42.4.2 Parallel input interface

The parallel interface is a flexible pixel interface that clocks in a bus of 8- to 16-bit in parallel, with an externally provided clock, and adds vertical and horizontal synchronization, provided either with specific pins (external sync) or flagged via specific data (as in CCIR601).

Interface (all inputs)

- Clk: clock up to 120 MHz, internally named dcmipp_pxclk
- Data: 16 IO pins, up to 120 Mbps/pin, 8- to 16-bit, pixels potentially serialized
- HSync, VSync, but no enable (EN)

Features

- Horizontal and vertical synchronization:
  - External, via the HSync and VSync pins.
  - Internal, via the CCIR601 flagging.

  Data provided during the horizontal or vertical blanking cannot be captured.
- Input pixel format (see *Section 42.3.6*)
  - Binary stream like JPG
  - Single cycle data:
    - Raw Bayer and monochrome: 8/10/12/14 bits in parallel
    - YUV422, RGB565:16 bpp on 16 bits in parallel
  - Double-cycle data:
    - YUV422, RGB565:16 bpp on 8 bits in parallel
    - RGB888/YUV444: 24 bpp on 12 bits in parallel
  - Triple-cycle data:
    - RGB888/YUV444: 24 bpp on 8 bits in parallel
- Flexible input:
  - MSB-vs-LSB bits can be swapped to ease PCB placement
  - Cycle0-vs-Cycle1 can be swapped (for 2-cycle long pixels).
  - Red-vs-blue components can be swapped (for RGB, or U-vs-V for YUV), this last swap operates only on the pixel pipes and not the dump pipe.

Hardware synchronization mode

In this mode two synchronization signals (DCMIPP_HSYNC and DCMIPP_VSYNC) are used.

Depending on the camera module/mode, data can be transmitted during horizontal/vertical synchronization periods. DCMIPP_HSYNC/DCMIPP_VSYNC act as blanking signals, as all data received during DCMIPP_HSYNC/DCMIPP_VSYNC active periods are ignored.

To correctly transfer images in the RAM buffer, data transfer is synchronized with the DCMIPP_HSYNC/DCMIPP_VSYNC signals. When the hardware synchronization mode is selected, and capture is enabled (CPTREQ bit set in DCMIPP_PxFCTCR), data transfer is synchronized with the deassertion of the DCMIPP_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by the IP-Plug to successive buffers or the same/circular buffer.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using embedded 32-bit codes, using the 0x00/0xFF values not used in data anymore.There are four types of codes, all with 0xFF00 00XY format. The embedded synchronization codes are supported only in 8-bit parallel data capture (in the DCMIPP_PRCR register, the EDM[2:0] bits must be programmed to “000”). For other data widths, this mode generates unpredictable results.

Note: Camera modules generate up to eight synchronization codes when in interleaved mode, while the DCMIPP reacts to one single code. As a consequence, an interleaved flow with an embedded synchronization has one every other frame discarded as not detected.

Mode 2

Four embedded codes signal the following events:

- Frame start (FS)
- Frame end (FE)
- Line start (LS)
- Line end (LE)

The XY values in the 0xFF00 00XY format of these codes are programmable (see *Section 42.14.9*).

A 0xFF value programmed as a frame end means that all the unused codes are interpreted as valid frame end codes.

In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

Mode 1

An alternative coding is the camera mode 1. This mode is ITU656 compatible.

The codes signal another set of events:

- SAV (active line) - Line start
- EAV (active line) - Line end
- SAV (blanking) - Start of line during interframe blanking period
- EAV (blanking) - End of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. User can therefore select a bit to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example:

- FS = 0xA5
- Unmask code for FS = 0x10

In this case the frame start code is embedded in bit 4 of the frame start code.

Note: FEC sequence must be sent before the transfer of the first frame, otherwise other codes are not decoded and the first frame can be lost after the DCMIPP has been enabled. After the first FE sequence, the following frame is captured based on FS sequence.

Error conditions

Error conditions can be detected when using the embedding synchronization data modes. Flags PRERRF in DCMIPP_CMSR2 and ERRF in DCMIPP_PRSR are used for this function. An interruption can be triggered based on error detection if bit PRERRIE in

DCMIPP_CMIER is set for the global interrupt (or bit ERRIE in DCMIPP_PRIER for a local interrupt line).

For instance, such kind of wrong sequences or values generate an error:

- FF 00 00 XY sequence detected with XY value different from the one set for FS/LS/FE/LE
- LS = 0xFF or LE = 0xFF
- If an LS or FS sequence follows an LS sequence

Software configuration

Table 381. DCMIPP_PRCR bit function

Bit ID	Function	Comments
ENABLE	Enables the parallel interface	Low-power mode when disabled.
PCKPOL	Defines clock polarity	– 0: Falling edge – 1: Rising edge
ESS	Defines if VSync and HSync synchronizations are provided by sideband (specific hardware IO) or by in-band (specific embedded codes) signals	– 0: Hardware (external) synchronization: using DCMIPP_HSYNC and DCMIPP_VSYNC pins. – 1: Embedded (internal) synchronization: using codes FSC, FEC, LSC, LEC and masks in registers DCMIPP_PRESCR and DCMIPP_PRESURy.
VSPOL, HSPOL	Defines the polarity of the IO DCMIPP_VSYNC and DCMIPP_HSYNC used for, respectively, the vertical and horizontal synchronization	– 0: Active low – 1: Active high
FORMAT	Defines the pixel format used on the IO DCMIPP_Dx	From 0 to 15
EDM	Defines the amount of valid bits received per cycle	8/10/12/14/16 bits
	The duration of a pixel is extracted by the bit-per-pixel (given by FORMAT) and bit-per-cycle (given by EDM) ratios	1, 2 or 3 cycles
SWAPBITS	Swaps the 16 bits of the parallel interface	15:0 becomes 0:15
SWAPCYCLES	Swaps, when a pixel is received in two cycles, the data from the first cycle with the data from the second cycle	Software must not activate it when a pixel is received in one or three cycles.

The embedded synchronization codes are configured by means of DCMIPP_PRESCR and DCMIPP_PRESUR registers (see *Table 382*).

Table 382. DCMIPP_PRESCR and DCMIPP_PRESUR bit function

Bit ID	Function
FSC, FEC	Frame-start and frame-end synchronization, when embedded synchronization ESS = 1
FSU, FEU	Masks for FSC and FEC codes

Table 382. DCMIPP_PRESCR and DCMIPP_PRESUR bit function (continued)

Bit ID	Function
LSC, LEC	Line-start and line-end synchronization, when embedded synchronization ESS = 1
LSU, LEU	Masks for LSC and LEC codes

42.4.3 Interface from CSI-2 host

The CSI-2 host and DPHY_RX (both external to the DCMIPP) provide the data from the CSI-2 camera sensor, forward them into the DCMIPP using the ISB byte header interface.

The ISB byte header features are:

- CSI-2 clock interface aligned with the DCMIPP processing clock (333 MHz)
- Data width: 32 bit (data and CSI headers are interleaved)
  - Frame synchronization short packet
  - Frame number, Data ID, and packet type are provided (see *Section 42.7.8*).
- Line Sync short packets are not received.
- Headers of data long packet:
  - Packet length, Data ID, and packet type are provided (see *Section 42.7.8*).
- Payloads of data long packet:
  - Data as 32-bit words, as collected by the CSI-2 host.
- Packet CRC is not received.

42.4.4 Test pattern generator (TPG)

This is a continuous frame generator that replaces the need, for a camera sensor, to test the DCMIPP and the system. It is located at the head of the DCMIPP pipelines, aside of the parallel interface, or the interface from the CSI-Host.

When active, frames are generated indefinitely, with a deterministic pattern, at the desired resolution, pixel format, and with a potential lifeline to show an evolution from frame to frame.

Functionality

- Configurable frame resolution (working at the pixel clock), with a configurable vertical blanking.
- Configurable pixel format: RGB888, RGB565, Raw8/10/12/14, mono, or YUV.
- Configurable pattern: color bars or color squares.
- Potential remapping of the colors to gray scales or to the YUV422 color domain.
- Potential lifeline, evolving from frame to frame. The lifeline is a horizontal bar that progressively shifts down, frame after frame. Its height is 1/64 of the frame height. The pixels in the lifeline have their component values inverted bitwise: 0xFF becomes 0x00, and red (0xFF, 0x00, 0x00) becomes cyan (0x00, 0xFF, 0xFF). The lifeline progresses downward at a speed of 1 line per frame, and restarts back at the top of the frame.

The test pattern generator generates one pixel (visible or blanking) per clock cycle of clk_proc for all formats, except RGB888, generated every two clock cycles.

Software configuration

The software configuration registers are not shadowed as assumed constants.

Table 383. DCMIPP_CMTPGCR1 and DCMIPP_CMTPGCR2 bit function

Bit ID	Function
WIDTH	Width of the visible pixels to generate per frame, up to 16383 pixels wide. The TPG adds an additional horizontal blanking, 1/8 of the width (with a minimum of 16 pixels), and one additional sync cycle. If FORMAT is YUV, WIDTH is rounded to the lower 16.
HEIGHT	Height of the visible lines to generate per frame, up to 16383 pixels wide. The TPG adds an additional vertical blanking of VBL lines.
VBL	Vertical blanking at the bottom of the frame.
FORMAT	Pixel format generated, among RGB565, RGB888, RAW, Mono, YUV422.
RT	Raw Bayer type: the sequence of components (RGGB, GRBG, GBRG, BGGR)
YT	YUV422 type: whether BT.601 or 709, and whether full or reduced range.
PATTERN	Defines the pattern used to fill the frame: – Can be a color bar pattern, with 8 vertical color bars generated. The colors, from left to right, are: white, yellow, cyan, green, pink, red, blue, black. – Can be a color square pattern, with the frame built of adjacent in 256x256 color squares, with a color gradient across each pixel.
GSEN	Transforms the color of the pattern to a grey scale.
LFLEN	Activates the lifeline. When active, it becomes visible, and progresses indefinitely, frame after frame.
TPGEN	Activates the TPG.

After the TPG activation with ENABLE = 1, a frame is generated with the desired resolution (WIDTH, HEIGHT) and pixel format (FORMAT). If FORMAT selects a raw Bayer format, then TYPE selects the raw Bayer sequence of component.

The frame rate (FPS) is configured using the vertical blanking (VBL). One pixel (visible or in the horizontal or vertical blanking) is generated per clock cycle at the clk_proc clock (except for the RGB888, with one pixel every two clock cycle). The relation between these items is:

FPS = clk_proc / Resolution (with blanking)

= clk_proc / (WIDTH + max (16,WIDTH/8) + 1) x (HEIGHT + VBL)).

For a usecase with a large needed visible resolution (WIDTH, HEIGHT), a needed FPS, and a given constant clock frequency clk_proc, VBL is configured as follows:

VBL = (clk_proc / FPS / (WIDTH + max(16,WIDTH/8) + 1)) - HEIGHT

Note that the RGB888 FORMAT and its pixel rate are 2x lower than those for other formats:

- When RGB888 is captured on the dump pipe (P0), it is mandatory to configure RGB888 to read RGB888 on the P0 buffer. In that case, the pixel generation is 1 pix / 2 clk.
- For all the other cases, including RGB888 used on pixel pipes, it is recommended to set FORMAT = RGB565. With TPG configured as RGB565, the pixel is converted to

RGB888 later in the pipe, resulting in a similar result in the pixel pipe. This permits to benefit of the full pixel rate, 1 pix / clk.

Configuration example

To generate an VGA frame at 30 fps, based on a clk_proc clock of 200 MHz, and with a raw Bayer 10 pixel format, with the sequence RGGB, configure:

- WIDTH: 640
- HEIGHT: 480
- VBL: 8766 = 200M / 30 fps / (640 + 80 +1) - 480
- FORMAT: 0x2B (= raw10)
- RT: 0 (RGGB)
- YT: 0 (not used)
- PATTERN: 0 (color bar)
- GSEN: 0 (keep the colors)
- LFLEN: 0 (no lifeline for now)
- TPGEN: 1 (test pattern generator active)

42.4.5 Input selection

Input selection block is ahead of any pipe to select the camera sensor module interface connected to the DCMIPP. Two interfaces are supported:

- Parallel interface: up to 16-bit parallel interface with physical synchronization signals (DCMIPP_VSYNC/DCMIPP_HSYNC) or software embedded ones (as in CCIR601).
- MIPI CSI-2 interface: Host ISB byte header internal connection from the CSI2 host controller placed in front of the DCMIPP.

Both interfaces are exclusive. The selected one is applied to all the input pipes.

The camera sensor interface is selected by means of bit INSEL of the DCMIPP_CMCR register.

42.4.6 Flow selection

The module, present in each pipe, handles the flow control around the CSI2 camera sensor interface. For each pipe, the software needs to configure which virtual channel (VC) and which Datatypes (DT) are processed by the pipe.

For Pipe0 and Pipe1, multiple Datatypes can be selected whereas only one virtual channel can be defined for a frame. Configuration can be changed frame by frame if the application requires it, offering some flexibility.

For Pipe2, when the pipe is independent with PIPEDIFF = 1, only one virtual channel and only one data type by frame can be selected.

The software has to reconfigure the flow selection module to handle interleaved frames from a sensor if it uses virtual channels as a mean to make the differentiation, accepting to lose frames since there can be only one virtual channel active for a given frame (it decreases the FPS).

Multiple cameras data flow can be received simultaneously through the CSI2 ISB header bus. For instance, for a system having two different camera data flows, it is possible to redirect one virtual channel on a pipe and the other one on the other pipe. It can be

interesting to dump them on Pipe0, but only one frame over two since the flow selection block has to be reconfigured to switch from one VC to another one.

The flow selection module is used also to share some image processing features (demosaicing, color conversion, statistics, contrast enhancement, auto exposure, statistics removal and basic decimation) between Pipe1 and Pipe2, whatever the camera sensor interface solution (parallel interface or CSI2 interface): when bit PIPEDIFF in the DCMIPP_P1FSCR register is 0, or the input is on the parallel interface that provides unique flow, the Pipe2 samples pixels from the output of the ISP of Pipe1 (see *Figure 342*). PIPEDIFF must be reconfigured when Pipe1 and Pipe2 are disabled.

Software configuration

The following configurations must be set first, because static in the DCMIPP_PxFSCR registers:

Pipe0

- DTMODE[1:0]: Mode for CSI flow selection into that pipe
  - 0: only flow VC/DTIDA is forwarded in the pipe
  - 1: flows VC/DTIDA and/or VC/DTIDB are forwarded in the pipe
  - 2: all Datatypes of the selected virtual channel, except the VC/DTIDA or VC/DTIDB, are forwarded in the pipe
  - 3: all Datatypes of the selected virtual channel are forwarded in the pipe
- VC[1:0]: Virtual channel ID of the CSI flow to select for further processing on the pipe
- DTIDA[5:0]: Data type ID A to select in the CSI flow for further processing in the pipe
- DTIDB[5:0]: Data type ID B to select in the CSI flow for further processing in the pipe
- PIPEN: activation of the Pipes, to be set after the above configurations.

Pipe1

- DTMODE[1:0]: mode for CSI flow selection into that pipe
  - 0: only flow VC/DTIDA is forwarded in the pipe
  - 1: flows VC/DTIDA and VC/DTIDB are forwarded in the pipe
- VC[1:0]: virtual channel ID of the CSI flow to select for further processing on the pipe
- DTIDA[5:0]: data type ID A to select in the CSI flow for further processing in the pipe
- DTIDB[5:0]: data type ID B to select in the CSI flow for further processing in the pipe
- FDTFEN: when the DTIDx is detected depending on DTMODE, the software forces the data type format to the value set into FDTF[5:0] for the pixel pipe processing. It can be used, for instance, to address sensor 3D camera through specific bridge using CSI2 “user data type” to make distinction between left and right camera sensor (each camera has its own user data type value, but sharing the same pixel format).
- FDTF[5:0]: data type value to be forced
- PIPEDIFF: Pipe2 to take same input as Pipe1 or differentiated:
  - 0: Pipe1 and Pipe2 take the flow selection configuration of Pipe1. Pipe1 shares the image processing functions with Pipe2.
  - 1: Pipe1 and Pipe2 are considered as two independent pipes, without any shared configuration.
- PIPEN: activation of the pipes, to be set after the above configurations.

Pipe2

- VC[1:0]: Virtual channel ID of the CSI flow to select for further processing on the pipe.
- DTIDA[5:0]: Data type ID A to select in the CSI flow for further processing in the pipe
- FDTFEN: When the DTIDx is detected depending on DTMODE, the software forces the data type format to the value set into FDTF[5:0] for the pixel pipe processing. It can be used to address sensor 3D camera through specific bridge using CSI2 “user data type” to make distinction between left and right camera sensor for instance (each camera having its own user data type value, but sharing the same pixel format).
- FDTF[5:0]: Data type value to be forced
- PIPEN: activation of the pipes, to be set after the above configurations.

If PIPEDIFF = 0 in DCMIPP_P1FSCR, the configuration bits are directly those set in the DCMIPP_P1FSCR register.

The above bit fields are meaningful when PIPEDIFF = 1.

42.4.7 Frame counter

A frame counter is available for a tag purpose and counts all the frame received on the selected pipe.

The counter is 32-bit, read-only. It is active at least when bit PIPEN of the pipe to whom it is connected is enabled (PIPEN = 1).

It provides an (almost) unique frame number, with a loop time of 4.5 years (if 30 fps).

The counter can be incremented, depending on the PSFC in DCMIPP_CMCR, at the FrameStart event of the selected virtual channel, when a CSI-2 camera sensor is connected, or at the FrameStart event of the pipe when the camera sensor module is connected with parallel interface.

The frame counter is cleared by setting CFC in the DCMIPP_CMCR register.

Software configuration

The frame counter must be configured before it is used for the pipe. Software action(s) must be performed in the DCMIPP_CMCR register:

- Select on which pipe the frame counter is connected by configuring bit PSFC[1:0]
- Frame counter runs as soon as the selected pipe is active (PIPEN = 1)

Note: It is possible to have the Frame counter run on a pipe, while having the pipe not flowing any pixel: this is achieved with pipe enable active (PIPEN = 1) to have the frame counter run, but with CPTREQ = 0 to avoid any pixel flow.

42.4.8 Frame control

The module is replicated for each dump and pixel pipe. It handles the frame control and the capture of each pipe, grouping data into frames, and capturing them when requested.

Definitions

There are two notions of frame events

- Event used inside the hardware (provided by the input interfaces):
  - FrameStart: when a frame starts at the input of the pipes (at frame control)
    - > Parallel interface with VSync fall
    - > CSI-2 interface: SoF (Start-of-Frame) packet
  - FrameEnd: when a frame finishes at the input of pipes (at frame control)
    - > Parallel interface: VSync rise (End-of-Frame)
    - > CSI-2 interface: EoF (End-of-Frame) packet
- Events exposed to software via DCMIPP events and interrupts:
  - VSYNC
    - > When a frame starts, observed at the input of the pipes (at frame control)
    - > It indicates that the current frame has started and has sampled its shadow registers, and that the software can start to configure the next frame
    - > It is recommended to use that interrupt to trigger the software that reconfigures the DCMIPP.
  - FRAME END
    - > When a frame finishes, observed at the output of pipes (when the last pixel has been written on the AXI)
    - > It indicates that the current frame has finished writing all its data to memory, and that the application can immediately start using them
    - > It is recommended to use that interrupt to trigger the software that forwards the memory buffer to other applications

Features

- Frame delineation: allows to dump data with the required frame granularity.
  - Extracts, per pipe, the FrameStart and FrameEnd event, based on the input Start-of-Frame and End-of-Frame.
    - > The FrameStart immediately generates a VSYNC event.
    - > The FrameEnd ripples till the end of the pipe, at the pixel packer level, where it generates the FRAME event.
  - Based on the algorithm detailed in *Extraction algorithm for FrameStart and FrameEnd*.
- Capture mode: can capture a frame snapshot, or capture continuously.
- Capture rate: can capture all frames, or one out of 2, 4, 8. It is only active when the Capture mode is in Continuous mode.
- Frame status: stores infos about the last captured frame (PxLSTFRM, PxLSTLINE), re-sampled at the FrameEnd event.

Extraction algorithm for FrameStart and FrameEnd

The extraction of the FrameStart and FrameEnd is straightforward.

In CSI-2 mode, each pipe is handling a unique virtual channel ID by frame. In case of interleaved flow in input of the DCMIPP, the software can decide to:

- Dedicate each virtual channel on a single pipe in case of multiple pipes, each of them having its own virtual channel number selected
- Decide to decrease the FPS ratio by changing the pipe configuration swapping with the other virtual channel ID (mainly in case of single pipe)

The interlaced mode can be handled by the software in CSI-2 mode by changing the pipe configuration frame by frame.

The following usecases are supported and detailed in the figures below:

- Default single VC: straightforward behavior, support in both pixel and dump pipes (*Figure 344*)
- Overlapping VC (supported on a single pipe via the loss of one frame when switching)
  - It can be supported by affecting each VC on a dedicated pipe (*Figure 346*)
- Exclusive VC (supported in both pixel and dump pipes), see *Figure 345*:

The use cases with two exclusive VCs can be handled either using one single pipe, by reconfiguring it from frame to frame, or by splitting the flow on more than one pipe, each of them handling its own VC:

- each flow arrives exclusively in one pipe and is processed as an independent frame, with its Start-of-Frame and End-of-Frame
- FrameStart and FrameEnd are generated per VC flow.

Figure 344. VC flow processed by one pipe

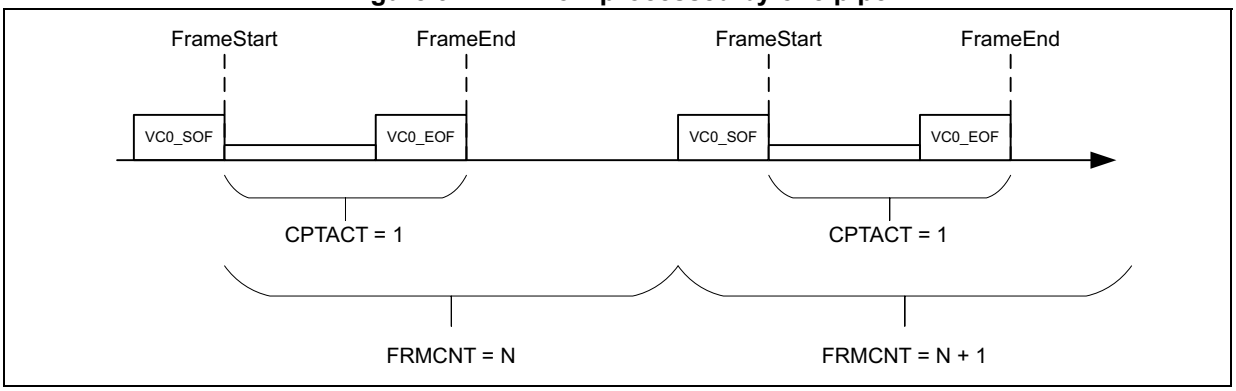
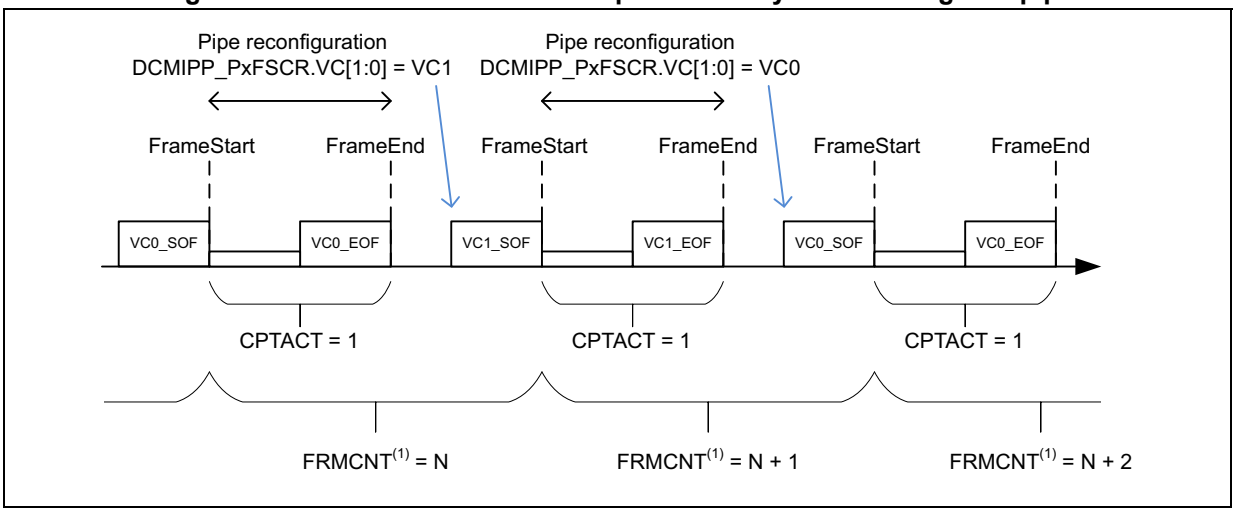
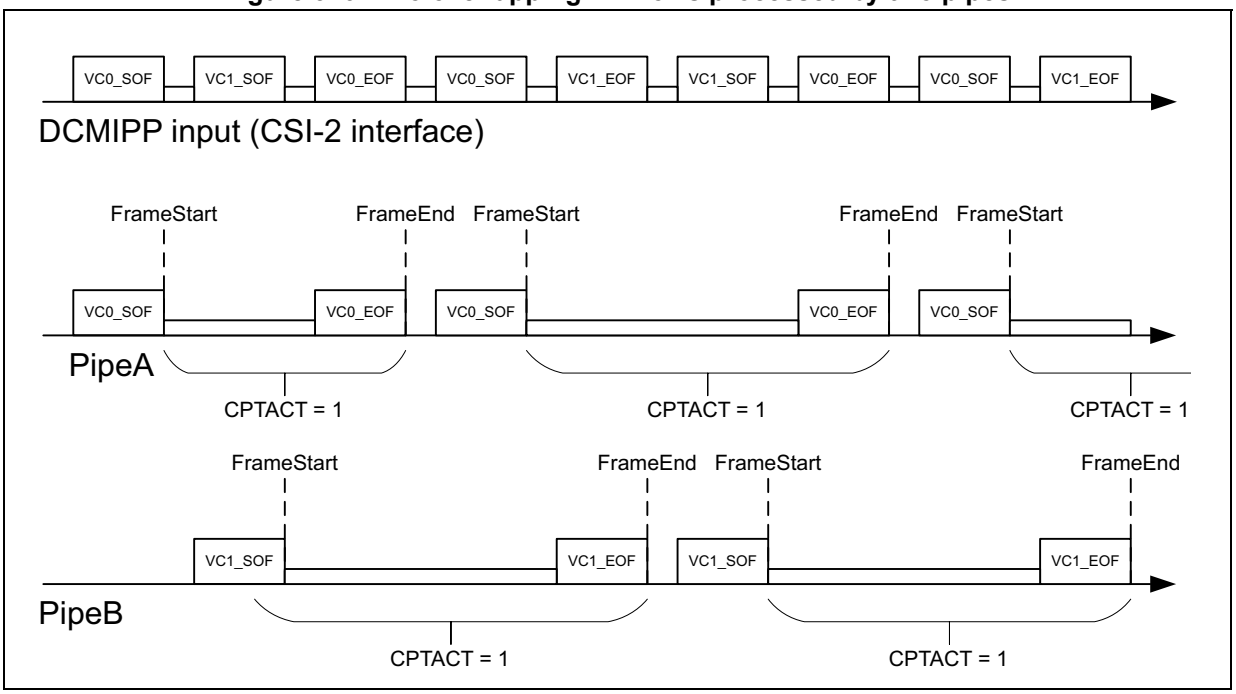


Figure 345. Two exclusive VC flows processed by one reconfigured pipe



1. If frame counter is attached to the pipe and if reset is not requested when changing VC.

Figure 346. Two overlapping VC flows processed by two pipes



Use cases

The Continuous mode is typically used to send to display or to software-analysis a continuous stream of frames.

The Snapshot mode is typically used to dump a high-resolution frame.

The Frame rate mode is typically used for a low-power continuous analysis, where only one frame every eight is dumped and analyzed.

The frame informations are typically used for interlaced video use cases, where the sensors transmit alternatively odd and even fields (top vs. bottom), and where software must be able to retrieve the type of field. Depending on the sensor, it can be retrieved from:

- the VC-ID, if the sensor uses one VC for one field and another for the other
- the LSB of the counted number of lines in that frame (indicates if the number is odd or even)
- the LSB of the frame counter extracted from the CSI information (if counting this pipe), which can be odd or even.

Software configuration

The frame control supports two different capture modes:

- Snapshot mode: the software captures single frames
- Continuous mode: the software captures a continuous sequence.

The timing diagram of these two modes is shown in *Figure 347*.

Figure 347. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes

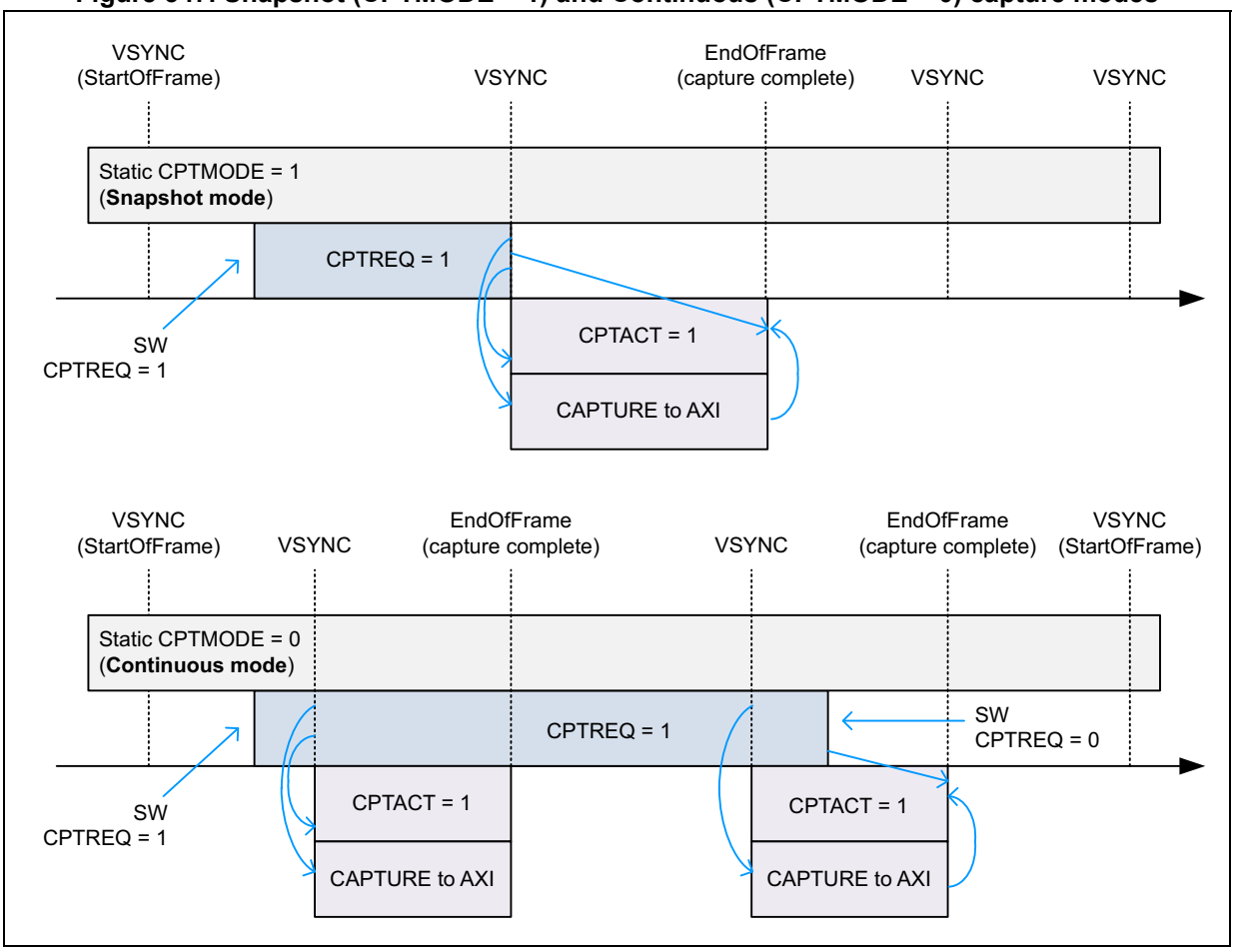


Table 384. DCMIPP_PxFCTCR bit function

Bit ID	Function	Values
CPTMODE	Capture mode	– 0: Continuous – 1: Snapshot
CPTREQ	Capture request	– 1: Capture requested
FRATE	Frame rate	– 0: Full rate capture – 1: 1/2-rate capture – 2: 1/4-rate capture – 3: 1/8-rate capture
CPTACT(1)	Capture status	– 1: Capture currently active

1. Bit read from the DCMIPP_PxSR register.

Configuration example

The software operations when using the Snapshot mode are the following:

1. Software sets CPTMODE = 1, to use the Snapshot mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the capture of one frame.
4. At the first following VSync, the HW samples CPTREQ at 1, with the following impact:
   a) The capture effectively starts: pixels flow into the pipe and are dumped in memory.
   b) CPTACT is set to 1, to mention that a capture is currently ongoing, and that it is best to not modify the configuration of the pipe operators, unless shadowed.
   c) CPTREQ is reset to 0, so that only a single frame is dumped.
5. At the following capture complete interrupt
   a) CPTACT is reset to 0, to signal that the capture is over.
   b) The capture is complete, no more pixels are flowing, hence software can restart to update the configuration of non-shadowed registers without any issue, and use the captured pixels in memory. Depending upon the sensor blanking, this period can be short.
6. At the next VSync, CPTREQ usually is sampled at 0, so that no more frames are captured
   – As soon as CPTREQ is reset to 0 the software can set it again to 1, to request a capture at the following frame. It means that a continuous sequence of frame can be captured by setting again CPTREQ to 1, continuously frame after frame.

The software operations when using the Continuous mode are the following:

1. Software sets CPTMODE = 0, to use the Continuous mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the continuous capture of frames.
4. At the first following VSync, CPTREQ is sampled at 1, with the following impact:
   – As in Snapshot mode, the capture starts, and CPTACT is set at 1.
   – Unlike the Snapshot mode, CPTREQ is not modified and remains at 1.
5. At the following capture complete interrupt:
   – As in Snapshot mode, CPTACT is reset to 0, capture stops, software can shortly reconfigure the pipes, and software can use the captured pixels.
6. At the next VSync, and as long as CPTREQ remains at 1, the capture restarts, similarly as described above.
7. To stop capturing later frames, software resets CPTREQ to 0.
   – An ongoing capture continues until completion.
8. At the next VSync, CPTREQ is sampled at 0, and the next capture does not restart.

42.4.9 Pipe deactivation

It is possible to abort a pipe to stop any frame acquisition and to potentially offer a mean to reprogram it completely (including non-shadowed registers), or to let it fully disabled for some time. The following operating mode has to be considered to correctly stop the pipe:

1. Disable the CPTREQ bit of the DCMIPP_PxFCTCR register of the corresponding pipe
2. Poll CPTACT = 0 status bit in the DCMIPP_CMSR1 register to check if the pipe is no longer active (idle state)
3. Disable PIPEN in the DCMIPP_PxFCTCR register.

When disabling the pipe, the last frame is processed completely by respecting the above operating mode.

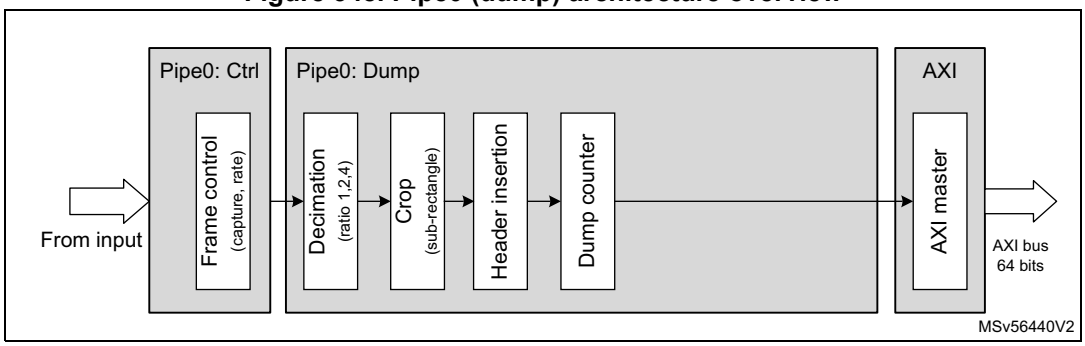
Note: To disable the parallel interface (PREN = 0 in the DCMIPP_PxFSCR), it is recommended to first disable the pipes considering the above operation mode, before switching off the parallel interface.

42.5 Pipe0 (dump pipe)

42.5.1 Overview

Pipe0 works as a dump pipe: it extracts data from the camera sensor module and dumps them (as-is) to the targeted memory, with some basic decimation and cropping 2D operations in between.

Figure 348. Pipe0 (dump) architecture overview



Pipe0 retrieves data from the camera sensor module connected to the DCMIPP through the 16-bit parallel interface or the CSI-2 interface, by means of the CSI-2 host controller peripheral. The selection between interfaces is managed by means of bit INSEL in the DCMIPP_CMCR register.

The flow selection chooses the virtual channels and data types to be processed by the pipe from the CSI2 host controller, see *Section 42.4.6* for more details.

The frame controller handles mainly the camera acquisition mode (continuous or snapshot, frame rate), see *Section 42.4.8* for more details.

When the input data from the camera is valid and supposed to be processed by Pipe0, 2D-cropping and decimation operations can be configured by the software. A dump counter combining with a limit amount of data to be set is offered to handle unknown length of data or to avoid the amount of data to be too wide within a frame.

Pipe0 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 42.13.2: Interrupts*).

It is also possible to write into the shadow registers within the processing of a frame. The values are loaded into the physical register based on the start of frame event.

42.5.2 Decimation

The decimation allows to cheaply downsize a frame.

Based on parallel interface camera module:

- Factor 1x, 2x, 4x at data/byte level depending on the configuration of bit field BSM[1:0] of the DCMIPP_P0PPCR register. The decimation is not working directly at pixel level, but it is based on either byte or 16-bit word granularity (for padded raw 10, for instance) to make equivalent decimation to one pixel out of two. Bit OEBS is used to start from the odd or even byte (or data) to capture first.
- The decimation can also be based on entire line, by capturing all the lines, or one line out of two thanks to bit LSM. Here, again, choice is given to the software to reject odd or even lines within the frame by configuring bit OELS.

When the camera sensor module is designed on CSI-2 interface, the decimation can be based filter out potentially, one line out of two. Indeed, each packet is by protocol composed for an entire line.

Note: Care must be taken when the CSI-2 header is dumped, as it is 32-bit wide, and it increases by one unit the data counter, taking part of the decimation. It is advised not to use the CSI header option when decimation takes place.

Table 385. DCMIPP_P0PPCR bit function

Bit ID	Function
BSM[1:0]	To select how much byte/data have to be captured within a line. This feature is available only for a parallel interface camera module.
OEBS	Allows the user to chose if the filtering starts from the odd or the even byte.
LSM	Possibility to filter out one line out of two.
OELS	The software can select if the filtering rejects the odd or even lines.

42.5.3 Crop/statistics selection/suppression

The dump pipe (Pipe0) has the capability to handle 2-D crop processing. The vertical cropping is based on line, thanks to HSYNC event (from the physical IOs or the embedded code detection if selected). It can be extracted from the CSI-2 packet information (each packet is one line wide by protocol).

The horizontal area on which the crop is applied is based on data (32-bit wide), and not on pixels since this pipe is not directly considering the pixels. It handles the data flow with 32-bit granularity. The CROP functionality is not supported when the pipe is convoying JPEG format. The ENABLE bit must be kept cleared into DCMIPP_P0SCSZR to avoid any unpredictable behavior.

The area to be captured within the frame has to be specified configuring the registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR. The starting point on the two axis are set accordingly as well as the width in both directions. It is possible to take the data inside or outside this area by means of bit POSNEG in the DCMIPP_P0SCSZR register. Some sensors can send sensor configuration data and statistics (histogram) on the very first and last lines of a frame. By specifying active the area outside of this window selection (POSNEG = 1), only statistic data are extracted by the way, and the software can decide to apply some processing on the data to correct, such as contrast or exposure.

Features

- 2D-Crop operation with granularity:
  - Lines in the vertical axis
  - 32-bit data in the horizontal axis
- Possibility to capture data inside or outside the defined window (valid image data or statistic data)
- Cropping with 0 value for HSIZE[11:0] or VSIZE[11:0] has no crop effect to the dimension for which the 0 value has been applied (vertical or horizontal axis).

Software configuration

Registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR have to be configured to set the crop feature into the dump pipe:

1. Configure the cropped horizontal starting point with HSTART[11:0] and the width with HSIZE[11:0] (both with a 32-bit data granularity).
2. Configure the cropped lines, starting at line VSTART[11:0] and with a height of VSIZE[11:0] (both with a line granularity)
3. If any value HSIZE or VSIZE is set to 0, the hardware does not consider crop operation in the vertical or horizontal dimension for which the 0 value has been applied, so that all the pixels in that dimension are sampled.
4. Select the inner or the outer part of the window for the data capture by configuring POSNEG bit.
5. Enable the crop feature setting ENABLE bit. This bit must be cleared when JPEG is selected as the input format for Pipe0.

Note: Cropping out the picture size (with too large HSTART or VSTART) leads to a not guaranteed processed frame.

42.5.4 Header insertion

The module works on the dump pipe(s) and it optionally inserts an header before the data of a former CSI packet.

Features

- Optionally inserts non-data information to help software interpreting the following data payload. The non-data info is Frame sync short packets and Header of data long packets (both non-data info into a 32-bit word), mapped as follows:
  - Frame sync short packet: defined by Frame type = SOF or EOF
    - > bits 31:16: Frame number
    - > bits 15:8: Reserved (forced to 0)
    - > bits 7:4: Virtual channel ID in CSI-2_v1.3, bits 7:6 = 00
    - > bits 3:0: Frame type (SOF = 0x1 or EOF = 0x2)
  - Header of data long packet (defined by below Frame type = DATA)
    - > bits 31:16: Amount of bytes in the payload
    - > bits 15:8: Data type ID in CSI-2_v1.3, bits 15:14 = 00
    - > bits 7:4: Virtual channel ID in CSI-2_v1.3, bits 7:6 = 00
    - > bits 3:0: Frame type (DATA = 0x0)

To keep consistency between the amount of bytes provided by the header versus those effectively dumped, it is recommended to not activate the header insertion simultaneously with the crop or decimation, as these functions modify the amount of dumped data.

When header insertion is active, the FrameEnd packet is dumped as a header, and is considered as the unique part of an additional line, inducing an event.

Software configuration

It is configured with the DCMIPP_P0PPCR register:

- HEADEREN:
  - 0: No header insertion
  - 1: Insertion of the headers, aligned on 32 bits.

42.5.5 Dump counter

The dump counter is present on dump pipe. It is used to count the amount of data that are dumped in that frame, and to potentially limit the amount dumped if too large. It allows the software to know the size of dumped buffer, and to make sure that no dump is made out of a preallocated buffer.

It is specifically useful when dumping a content whose length is unknown prior to reception, like an encoded JPG stream. When dumping a pixel frame, the size is known thanks to the configured width and height of the frame.

The counter is counting 32-bit data for almost all the input formats, even if the value is expressed in number of bytes. The counter increment is 4-bytes granularity for all formats except the JPEG byte stream input mode, for which the counter granularity is 8-bit because the application does not know the amount of dumped data.

Features

- Dump counter, counting the amount of bytes dumped. It counts the amount of dumped 32-bit words (the increment is 4), except when connected to a parallel interface camera and dumping a 8-bit data in byte stream mode (see FORMAT bit field in the DCMIPP_PRCR register), where it counts per byte (the increment is 1).
  - The counter saturates at 0x03FF_FFFF.
  - At FrameEnd, the counter value is recopied in a readable shadow register and the counter itself is reset to start the count of the next frame.
- Dump limit, that clamps the dumped words below a maximum limit:
  - If the counter reaches the limit, all following bytes are deleted until the next VSync.
  - The counter continues to count over the limit, to indicate how many bytes have been received and must have been dumped if not deleted.
- Interrupt limit is triggered when some bytes are deleted due to dump limit crossed.
- CSI headers, when dumped (HEADEREN = 1), participate to the counted and limited data volumes.
- Monochrome or raw Bayer padding (alignment of 10/12/14 bpp to 16 bpp) is also taken into account in data volumes.

Software configuration

Table 386. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function

Bit ID	Function
ENABLE	1: Enables the limit check.
CNT	Read-only, counts the amount of data with a 4-byte granularity for most of the input formats or with an 8-bit data granularity in JPEG byte stream mode selected into DCMIPP_PRCR.FORMAT). The counter increment is based on dumped data at the pipe output standpoint. The value is expressed in bytes, whatever the input format.
LIMIT	Gives the maximum amount of 32-bit data that can be dumped. Limit value 0 is inconsistent and the processed frame is not guaranteed if ENABLE bit is set.
LIMITIE	DCMIPP_PxIER.LIMITIE bit enables the interrupt.

Note: The dump counter sees the CSI-2 packet headers as normal 32-bit words. Raw Bayer and monochrome pixels on 10/12/14 bpc are padded onto 16 bits, and are thus slightly larger than their 10/12/14 bit size.

42.6 Pipe1 (ISP part)

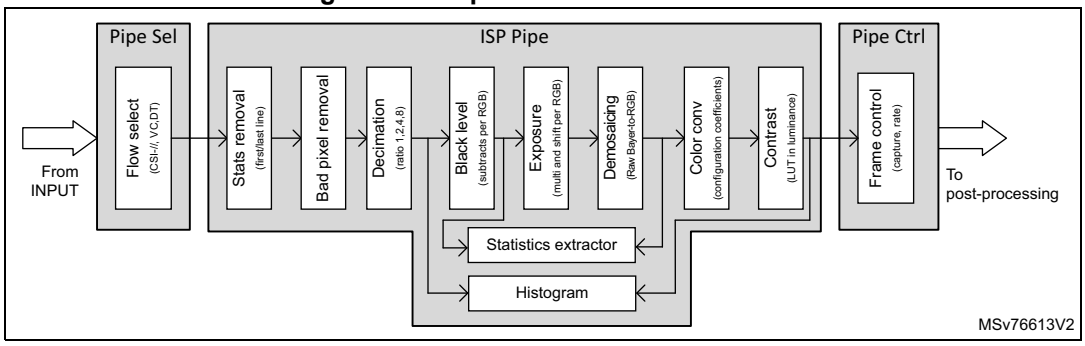
42.6.1 Overview

Pipe1 is a pixel pipeline in which some image processing (ISP) and post-processing functions are integrated, to relieve the software from very specific (time and memory consuming) operations. The ISP part implements functions like:

- Statistics removal within the dataflow
- Bad pixel detection to automatically remove noisy pixels
- Basic decimation to reduce high resolution camera sensor to fit with the image processing maximum resolution tolerance
- Auto exposure and statistics extraction based on pixel analysis
- Demosaicing algorithm to convert raw Bayer input pixels to RGB ones
- Color conversion and contrast enhancement functions

Pipe1 ISP block diagram is shown in *Figure 349*.

Figure 349. Pipe1 ISP architecture view



Pipe1 is the most featured pixel pipeline, even if the image processing functions can be shared with Pipe2 (refer to *Section 42.8.2*).

Pipe1 retrieves data from the camera sensor module connected to the DCMIPP with the 16-bit parallel interface or with the CSI-2 interface, thanks to the external CSI-2 host controller peripheral. The selection between interfaces is managed by means of bit INSEL in the DCMIPP_CMCR register.

The flow selection chooses the virtual channels and data types to be processed by the pipe from the CSI2 host controller. Refer to *Section 42.4.6: Flow selection* for more details.

The frame controller handles mainly the camera acquisition mode (continuous or snapshot, the frame rate). Refer to *Section 42.4.8: Frame control*.

Pipe1 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 42.13.2: Interrupts*).

It is also possible to write into the shadow registers during the processing of a frame. The values are loaded into the physical register based on the following start of frame event.

42.6.2 Byte-to-pixel conversion

The module works on the pixel pipes: it gets 32-bit words from the input (parallel or CSI) interfaces, extracts the pixels depending on the provided pixel format, and maps them onto the triple-component pixel pipeline, that is 14 bpc at this stage (and 8 bpc later downstream).

Functionality

- Extracts various input pixel formats to a triple R, G, B lane
  - All CSI-2 formats of versions CSI v1.3 and v2.0 are supported, except Raw 6/7 bpc, YUV422-10 bpc, YUV420 (see *Table 379*)
- YUV420-to-YUV444 conversion is not supported
  - Pipe1 and Pipe2 cannot extract and process YUV420 pixels received from the CSI interface
- YUV422-to-YUV444 conversion:
  - Done by replication of the U and V components. It thus assumes that the location of the common chroma is defined at mid-distance between the two pixels.
- Extension of short triple component (RGB/YUV) to the output triple 8 bpc:
  - The extension is done by replicating the input MSB onto the missing output LSB. The replication onto the LSB allows to map any input dynamic on the full downstream dynamic.
  - Examples of 5 and 6 bpc extended to the output 8 bpc:
    - > Input Di[4:0]: Output Do = Di[4:0] and Di[4:2]
    - > Input Di[5:0]: Output Do = Di[5:0] and Di[5:4]
- Keeps as-is the single-components, usually wide:
  - The monochrome and raw Bayer 10/12/14 bpc are fed into the pipes as-is (the monochrome assumes a Y, with U = V = 128).

Software configuration

The used pixel format depends upon the source of the pixels:

- CSI-2 interface: the pixel format is embedded in the received CSI-2 packets, and defined in the data type field of the packet header.
  - When FDTFEN = 0, the byte-to-pixel conversion uses the data type provided by the CSI-2 packet, and works automatically, without need for any configuration.
  - When FDTFEN = 1, the data type is configured by software in the FDTF field.
- Parallel interface: the pixel format is determined by the source of the pixels, i.e. the camera sensor. The byte-to-pixel extracts the pixels based on the pixel format provided by the software in the parallel interface configuration, in the field of the DCMIPP_PRCR register.

42.6.3 Statistics removal

The module is used to removing potential statistic data inserted by a sensor, prior to starting the demosaicing conversion that is multi-pixel and would be impacted by remaining statistics. It is typically used by sensors with a parallel interface, which does not have any way to distinguish between statistics and pixel data. CSI-2 sensors typically split these flows using different virtual channel IDs.

Features

- Suppression of some of the first lines of the frame.
- Suppression of the last lines of the frame, after a configured line N.

Software configuration

Table 387. DCMIPP_P1SRCR bit function

Bit ID	Values	Comments
CROPEN	1: Active	If CROPEN = 0 all pixels are fed through.
FIRSTLINEDEL	0 to 7	Amount of first lines to delete when CROPEN = 1
LASTLINE	0 to 4094	Amount of lines to keep after the removed first lines. If LASTLINE is null, all remaining lines after FIRSTLINEDEL are fed through.

Configuration example

If the received frame contains four lines of statistics, followed by 100 lines of pixels and eight lines of ancillary data, the configuration to extract the 100 lines of pixels is the following:

- CROPEN = 1, to have the removal active
- FIRSTLINEDEL = 4, to remove the first four lines
- LASTLINE = 100, to keep the next 100 lines

42.6.4 Bad pixel removal

This module aims at detecting and correcting artifacts generated by a bad pixel on the sensor array. It is based on a correlation extraction, and not on fixed locations.

Functionality

- Works for raw Bayer flows only, as RGB flows are assumed as already processed.
- Scans all the arriving components, and extracts the horizontal continuity correlation versus the two left and two right neighbors, on both the local component and on the adjacent raw Bayer component:
  - if a component is seen as too light, it is declared as bad
  - too dark components are not extracted, to avoid false positives
- A component assumed bad is replaced by the horizontal average of its two neighbors (with the same component).

Software configuration

Table 388. DCMIPP_P1BPRCR and DCMIPP_P1BPRSR bit function

Bit ID	Function
ENABLE	– 1: Active – 0: No component checked nor replaced. To be enabled only for raw Bayer flows, as it corrupts RGB flows.
STRENGTH	3 bits, 0 (high tolerance) to 7 (aggressive detection)
BADCNT	12-bit counter, reporting the amount of bad components corrected. The counter saturates at its maximum value (4095). At FrameEnd, the counter is re-clocked (the read value remains stable one full frame), even if the frame is skipped (not captured), and reset to restart accumulation at the next frame.

Configuration example

A too aggressive bad pixel detection might detect false bad pixels and erroneously correct them (it can occur for very thin light vertical lines that cannot be distinguished from bad pixels in the implemented horizontal filter). To avoid such artifacts, it is advised to setup a software loop-back that tunes the filter strength so that the detected bad pixels remain within a predefined valid range.

Start configuration (or without software feedback loop):

- ENABLE = 1 (enabled)
- STRENGTH = 0 (high tolerance)

Software feedback loop:

Assumes a maximum amount of 100 bad pixels on the array (any other value can be used).

- BADCNT read: gets the amount of corrected bad pixels of the last frame
- Low-pass filter on the BADCNT, and/or hysteresis
  - BADCNT > 100: too aggressive, decrease STRENGTH
  - BADCNT < 30: too tolerant, increase STRENGTH (assuming there are bad pixel on the array)

Note: The expected amount of bad pixels (100 in the above example) can be set to other values, or made dependent upon the equivalent gain of the sensor (analog gain, exposure duration). A high-gain picture has more noisy pixels and can accept a more aggressive filter to smooth the noise, while a low-gain picture has mostly correct pixels, only very uncorrelated pixels need to be corrected.

42.6.5 Input decimation

The decimation makes it possible to cheaply downsize a frame, by a factor 2, 4, or 8, in both horizontal and vertical directions. It supports a special mode adapted to a raw Bayer format in input.

The decimation is redundant with the downsize, that brings a higher quality (the decimation discards the data of the dropped pixels, while the downsize averages all pixels and does not lose information).

The decimation has two main usages:

- Reduces the width of very high resolution sensors to one accepted by the raw Bayer-to-RGB conversion. Indeed, the raw Bayer-to-RGB embeds a line buffer that limits its processed images to a maximum width of 2688 pixels. The decimation makes possible, for instance, to plug a 12 Mpixel sensor (4000 x 3000), decimate its width by a factor 2x to 2000 pixels wide, and have it converted to RGB.
- Complements the downsize (limited to ratio of 8 x 8), to have a combined downsizing ratio of 64 x 64. It is recommended to have the largest ratio implemented in the downsize, and the decimation complement to that. For instance, for a ratio 30x, is it recommended to have a ratio 4x in decimation, and a ratio 30 / 4 = 7.5 in the downsize.

Features

- Forwards one item every 1, 2, 4, 8 pixels, independently, vertical and horizontal.
- Supports raw Bayer formats, where RG-GB components must remain grouped. In that mode the above forwarded items are not pixels, but a double component (for example R and G or G and B for a RGGB format), in both vertical and horizontal directions. For instance, with a decimation of 2x, the input sequence R0, G0, R1, G1, R2, G2, R3, G3 gets forwarded as R0, G0, R2, G2 and not R0, R1, R2. The same applies vertically.

Software configuration

Table 389. DCMIPP_P1DECR bit function

Bit ID	Function	Value	Comments
HDEC[1:0]	Ratio (2 bits)	0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8
VDEC[1:0]		0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8

The raw Bayer mode in decimation is indirectly activated when the Datatype is defined as raw Bayer.

42.6.6 Black level calibration

This module suppresses a potential positive black level offset, which happens when the sensor sends non-null pixels when capturing a black picture.

Functionality

- Subtracts an offset from the R, G, B components.
- A different constant is used for R vs. G vs. B, including within raw Bayer pixels.

Software configuration

Table 390. DCMIPP_P1BLCCR bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
BLCR	-	8-bit offset, for red
BLCG	-	8-bit offset, for green
BLCB	-	8-bit offset, for blue

The offset subtracted is aligned components MSB: for instance, if components are defined on 12 bits, the 8-bit offset subtracted for red is BLCR « 4.

Configuration example

It is possible to use the statistics extraction, to extract the long-term most-dark R, G, B components, and assign their darkness as black. Their RGB values become the black level offset for each RGB.

For a raw Bayer sensor with 12-bit components, if the measured black level offset for red is 64 (out of a maximum of 4095), configure BLCR as 16.

42.6.7 Exposure compensation and white-balance calibration

This module aims at compensating bad exposure (too short/dark or too long/light), and at compensating a bad color balance to recenter the colors to more white ones.

The exposure and white balance operations are purely linear, and do not distort colors if performed correctly.

Functionality

- Multiplies R, G, B by a configured floating-point constant.
- A different constant is used for R vs. G vs. B, including within raw Bayer pixels.

Software configuration

Table 391. DCMIPP_P1EXCR1 and DCMIPP_P1EXCR2 bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
MULTR, SHFR	-	8-bit multiplier (mantissa), and up-shift value (0, 1, 2, 3, 4, 5, 6, 7) for red components.
MULTG, SHTG	-	8-bit multiplier and exponent, for green components.
MULTB, SHTB	-	8-bit multiplier and exponent, for blue components.

The idle setting (that sends out a similar value as input) is:

- MULTR = MULTG = MULTB = 128
- SHFR = SHFG = SHFB = 0

Similar idle couples are MULTx = 64 and SHFx = 1, or 32 and 2, or 16 and 3.

Configuration example for exposure alone

It is recommended to use the exposure control in conjunction with the contrast enhancement below in the pipe: the idea is to deliberately keep an exposure lower than ideal, to avoid burning the whites: the contrast enhancement, thanks to non-linear operator on luminance, lightens the dark pixels, and reduces the dynamic of the light pixels without burning them.

It is recommended to use the statistics to extract the average luminance across the whole frame (done best by extracting the R, G, B averages, and computing the luminance from the RGB).

The idea is to push the average output luminance to 128, or if working with the contrast enhancement too, to push it to a lower value, for instance 64.

When measuring an average luminance of 26 (target is 64), the exposure factor to apply is 64 / 26 = 2.5x = 1.25x * 2, hence the following configuration must be set:

- MULTR = MULTG = MULTB = 128 * 1.25 = 160
- SHFR = SHFG = SHFB = 1

Configuration example for an additional white balance

An example of easy way to recenter the colors is using the Gray-world algorithm.

This algorithm assumes that a default scene has an inherent balance of its red, green, and blue pixels, so that the average values of the red, green, and blue components across the ideal picture are the same. If the measured averages are not the same, it means that there has been a distortion of the white balance, that must be corrected by amplifying or reducing the red, green, and blue strengths in the exposure module, thanks to the differentiated red vs. green vs. blue multipliers and exponents.

In practice this is the same approach used for the exposure compensation, but, instead of forcing the output luminance to a default target (64 in the example), user must force the red, green, and blue averages towards the same exposure target.

As an example:

- Exposure target is defined as 64.
- Measured red is 26: red factor is 64 / 26 = 2.5x, hence MULTR = 160, SHFR = 1.
- Measured green is 32: green factor is 64 / 32 = 2.0x, hence MULTG = 128, SHFG = 1.
- Measured blue is 10: blue factor is 64 / 10 = 6.4x, hence MULTB = 205, SHFB = 2.

42.6.8 Demosaicing

The module works on the pixel pipes, and implements a mid-quality conversion from raw Bayer 8 bpp to an RGB 24 bpp, with no implicit half-sizing: the output RGB pixel rate is the same as the input component rate.

The conversion targets real-time but medium quality purposes, like display or low-resolution software analysis. For higher quality conversions, it is recommended to convert the raw Bayer in software, with higher quality algorithms but limited to non-real-time, thus still-picture applications.

The algorithm takes the surrounding 3 x 3 components in input, applies linear heuristics to determine the most likely missing components, and generates an output RGB888 pixel.

To provide the 3 x 3 input matrix to the algorithm, the demosaicing module embeds a double line-buffer, 2 x 2688 components wide. As such, it limits the maximum width of a converted demosaicing buffer to 2688 components/pixels wide. Wider camera sensor resolution can be connected and goes through the first decimation stage in front of the demosaicing operation.

Note: This demosaicing conversion algorithm can create some artifacts, like Zipper. Such artifacts are much reduced by a later downsizing or even suppressed for a 2 x 2 downsizing. Applications like display, video encode, or software analysis usually require such a downsize, so the artifacts become invisible.

The underlying heuristic filters try to extract and adapt to edges, horizontal and vertical lines, and lone pixels.

Functionality

- Input pixel format: from 1 (raw Bayer) to three (RGB/YUV) components on 8 bits
- Max input width of 2688 components
- Demosaicing conversion:
  - 2-line buffer
  - provides a 3 x 3 matrix to the algorithm + one trailing component (2 left)
  - Extraction and optimization for specific shapes:
    - Edges (on left, right, top, bottom)
    - Lines (horizontal and vertical)
    - Peaking (lone pixels detection)
- Output pixel format: three RGB components on 8 bits.

Software configuration

Table 392. DCMIPP_P1DMCR bit function

Bit ID	Values	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components.	If the camera sensor transmits an RGB or YUV frame and not raw Bayer, the demosaicing conversion must be disabled, by setting Enable = 0. All other parameters are ignored.
TYPE	– 0:RGGB – 1:GRBG – 2:GBRG – 3:BGGR	Default raw Bayer with three components (R, G, B)
EDGE	– 0: No edge detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
LINEH	– 0: No horizontal line detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
LINEV	– 0: No vertical line detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
PEAK	– 0: No peak detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-

The four filters, EDGE, LINEH, LINEV, PEAK, improve the zoomed accuracy of the filtered picture (thus mostly luminance correctness, like no Zipper artifact), however at the expense of some chrominance blurring or distortion. Some scenes can require specific settings:

- Scenes where chrominance distortion is very visible (like a mostly black and white scene viewed by an RGB sensor) can require lower strength of the filters.
- On the opposite, scenes heavily loaded with geometric shapes (and thus source of artifacts) can require strengthening the filters.

Globally, 6/4/4/2 is a good compromise for the filter strength.

Configuration example

If the camera sensor is raw Bayer, with a grid with a red top-left pixel, the next configuration can be used:

- ENABLE = 1 to have the demosaicing active
- TYPE = 0 to have a top-red pixel, thus a RGGB grid
- EDGE, LINEH, LINEV, PEAK = (4, 4, 4, 4) as typical setting (so an internal strength of 8 for each parameter).

42.6.9 Color conversion

This module implements a color conversion with flexible (fully programmable coefficients), which can be used to map the following features:

- YUV-to-RGB: when the pixels of a YUV-sensor are used as RGB (display, GPU)
- RGB-to-YUV: when the pixels of a RGB-sensor are used by software (for example video encode)
- Tone change: to map special color references (sepia), emphasize some tone/colors
- Monochrome: to compute a monochrome from RGB/YUV with a specific linear mix
- Expansion: to handle a YUV (or RGB) expansion or compression
- Gamma: to emulate a gamma correction via a linear scale

Features

- Linear rotation of color space: 3 x 3 matrix + 3 x 1 column.
- Clamping optional on either [16 to 235] for Y, R, G, B and [16 to 240] for U, V.

Software configuration

Table 393. DCMIPP_PxCCyy (yy = R0, R1, G0, G1, B0, B1) bit function

Bit ID	Value	Comments
ENABLE	– 1: Activates the color conversion	-
CLAMP	– 1: Clamps the output pixel, when using a reduced dynamic	-
TYPE	– 1: Clamps as RGB – 0: Clamps as YUV	-
Matrix3x3_Coeff	-	11 bits (sign, 2 int, 8 dec, with sign as complement to 2)
Added3x1_Coeff	-	10 bits (sign, 9 int, with sign as complement to 2)

Configuration example

Table 394 provides examples of the most used conversions:

- To / from RGB from/to YUV (as SDTV ITU-R BT.601 or HDTV BT.709).
- With a full (0 to 255) or a reduced (16 to 235) dynamic for Y and RGB, 16 to 240 for CbCr.
- Matrix fields are defined as XY bit fields in the color conversion coefficient registers, with X being the input R or G or B, and Y being the output R or G or B. For conversions to/from YUV, the R must be assumed as Cr (= V), G must be assumed as Y, and B must be assumed as Cb (= U).
- Coefficients are defined signed, and multiplied by 256 when there are decimals (the matrix coefficients). Negative values are on 11 bits: N11(val) = XOR (val, 0x7FF) + 1, on 10 bits for Added_Coeff: N10(val) = XOR (val, 0x3FF) + 1.

Table 394. Color conversion: examples of coefficients

Conversion	Output components	Input components				
RGB-255 to YUV601-255	-	R	G	B	Added	Clamping
	Cr	131	-110	-21	128	[0 to 255]
	Y	77	150	29	0	[0 to 255]
	Cb	-44	-87	131	128	[0 to 255]
YUV601-255 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	351	256	0	-175	[0 to 255]
	G	-179	256	-86	132	[0 to 255]
	B	0	256	443	-222	[0 to 255]
RGB-255 to YUV709-255	-	R	G	B	Added	Clamping
	Cr	131	-119	-12	128	[0 to 255]
	Y	55	183	18	0	[0 to 255]
	Cb	-30	-101	131	128	[0 to 255]
YUV709-255 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	394	256	0	-197	[0 to 255]
	G	-118	256	-47	82	[0 to 255]
	B	0	256	456	-232	[0 to 255]
RGB-255 to YUV601-240	-	R	G	B	Added	Clamping
	Cr	112	-94	-18	128	[16 to 240]
	Y	66	129	25	16	[16 to 235]
	Cb	-38	-74	112	128	[16 to 240]
YUV601-240 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	409	298	0	-223	[0 to 255]
	G	-208	298	-100	135	[0 to 255]
	B	0	298	517	-277	[0 to 255]
RGB-255 to YUV709-240	-	R	G	B	Added	Clamping
	Cr	112	-102	-10	128	[16 to 240]
	Y	47	157	16	16	[16 to 235]
	Cb	-26	-87	112	128	[16 to 240]
YUV709-240 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	459	298	0	-248	[0 to 255]
	G	-137	298	-55	77	[0 to 255]
	B	0	298	541	-289	[0 to 255]

Table 394. Color conversion: examples of coefficients (continued)

Conversion	Output components	Input components				
RGB-240 to YUV601-240	-	R	G	B	Added	Clamping
	Cr	131	-110	-21	128	[16 to 240]
	Y	77	150	29	0	[16 to 235]
	Cb	-44	-87	131	128	[16 to 240]
YUV601-240 to RGB240	-	Cr	Y	Cb	Added	Clamping
	R	351	256	0	-175	[16 to 235]
	G	-179	256	-86	132	[16 to 235]
	B	0	256	443	-222	[16 to 235]
RGB-240 to YUV709-240	-	R	G	B	Added	Clamping
	Cr	131	-119	-12	128	[16 to 240]
	Y	55	183	18	0	[16 to 235]
	Cb	-30	-101	131	128	[16 to 240]
YUV709-240 to RGB240	-	Cr	Y	Cb	Added	Clamping
	R	394	256	0	-197	[16 to 235]
	G	-118	256	-47	82	[16 to 235]
	B	0	256	465	-232	[16 to 235]

42.6.10 Contrast enhancement

This module ends the image processing pipe. It makes possible to enhance the contrast of the frame, by non-linear emphasis of the luminance of the pixels, and to provide more luminance quantification steps at highly used luminances (usually central grays), by reducing the amount of quantification steps less used by the picture (usually the few very dark and very light pixels).

Functionality

- Inputs RGB from the pipe
- Computes the luminance of the local pixel
- Extracts the desired luminance increase/decrease factor, in function of the input luminance, non-linearly (using a look-up table)
- Multiplies the input RGB with the amplification factor
- Outputs the clamped RGB components on 8 bits

Software configuration

Table 395. DCMIPP_P1CTCR1,2,3 bit function

Bit ID	Values	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
LUMx (x = 0 to 8, respectively inputs at 0, 32, 64, 96, 128, 160, 192, 224, 256)	0 to 63	Amplification factor: – LUMx = 8 halves the pixel luminance – LUMx = 16 does not modify the pixel luminance – LUMx = 32 doubles the pixel luminance

Configuration example

There are multiple approaches to handle contrasts.

It is possible to flatten the spectrum of luminance density (that is, to try have a similar amount of pixel per each segment of luminance), which is to fill each of the eight segments of luminance with 12.5% of the pixels of the frame.

An intermediate approach is to proceed with only the upper and lower half pixels: half of pixels must have a luminance below and another half above the mid-dynamic (128 for 8-bit components).

It is advised to use the statistics extraction module, sample the luminance after the exposure control (to avoid an impact on the exposure compensation), and sort it in the 12 available bins (refer to *Section 42.6.11*). From the extracted bins the user must linearly extrapolate the luminance of the median of the amount of pixels. The luminance of median pixel must be increased/decreased to be moved to the targeted output median luminance, which is recommended to be the median dynamic (128).

For instance, a 400 kpixels VGA picture can have 200 kpixels below a luminance of 64, and the other 200 kpixels above that value. The ideal luminance amplification factor can be set to 128 / 64 = 2x:

- pixels darker than 64 are lightened by a factor 2x
- pixels lighter than 64 are lightened a bit less, decreasingly
- pixel of luminance of 255 get an idle, 1x, factor.

The above assertions can be used to compute the LUMx values to configure:

- Ratio 2x from lum = 0 to 64:
  - LUM0 = LUM1 = LUM2 = 32.
- Decreasing ratio from 32 to 256, best exponentially decreasing:
  - LUM3 = 27
  - LUM4 = 23
  - LUM5 = 20
  - LUM6 = 18
  - LUM7 = 17
  - LUM8 = 16 (the maximum luminance, 256, must remain idle).

42.6.11 Statistics extraction

This module extracts several statistics across the frame, to provide inputs to the software image compensation algorithms, that configure feedbacks to the following implemented hardware modules: black level calibration, exposure compensation, white balance, and contrast enhancement.

The statistics are accumulated during a full frame, and sampled at the end of the frame, thus allows the software to have the full next frame to read the accumulated statistics, process them, define the adapted configuration for the hardware feedback modules, and to configure these modules.

Thanks the shadowing mechanism, the newly configuration parameters are active at the following frame, thus active at an average two frames after the statistics are accumulated. The statistics extraction block is considering all frames selected by means of the DCMIPP_P1FSCR configuration registers. Statistics block is running even when the frame is not supposed to be captured, respect to the DCMIPP_FCTCR configuration register.

The latency is not an issue, as the captured scene is expected to be almost static, and typical software compensation algorithms filter the statistic variations to avoid oscillations and visual artifacts (the low-pass filter typically has a one-second period).

Functionality

- Three statistics accumulators available, allowing three statistics extractions per frame.
  - Different types of statistics can be extracted frame over frame
- 32-bit accumulators, allowing to accumulate 8-bit samples over 16 Mpixels.
- Average mode, that extracts averages (accumulates component values)
  - Capability to discard extreme values (if value is <16 or >240, same for 32 or 64).
- Binning mode, which extracts presences (accumulates 256 per fit)
  - 12 available bins (values: <4, <8, <16, <32, <64, <128, and reverse to 256).
- Capability to define a sub-rectangle within the frame where statistics are extracted
  - Allows to focus the statistics on a specific region of interest.
- Raw Bayer supported, with the statistics using the specific locally available R, G, B components of the raw Bayer
- Statistics can be performed on either the input data (before any pixel modification) or after demosaicing, to benefit of a full RGB and thus luminance presence per pixel.

Software configuration

Table 396. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
MODE	– 0: Average mode	Values of pixels are accumulated as-is.
	– 1: Bin mode	Values of pixels are used to see if the pixel fits one of the 12 bins

Table 396. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function (continued)

Bit ID		Value	Comments
SRC (3 bits)		– (R, G, B, L)_up	– Up: refers to sampling after black level. – L: luma (Y') computed as R / 4 + G / 2 + B / 4 when receiving RGB pixels, (R or G or B) when in raw Bayer mode.
		– (R, G, B, L)_down	– Down: refers to sampling after demosaicing (raw Bayer to RGB conversion), thus after both the exposure and white balance compensation, and when true RGB components (and thus luminance components) are available.
BINS (2 bits)	MODE = Average	– 0: AllPixels: no components are rejected – 1: NoExt16: components are rejected if < 16 or ≥ 240 – 2: NoExt32: components are rejected if < 32 or ≥ 224 – 3: NoExt64: components are rejected if < 64 or ≥ 192	Enables to reject extreme data, not accumulated for statistics. Additional statistics (with MODE = Bin) are needed to count the amount of selected pixels. BINS = 0 is likely to be the most used setting.
	MODE = Bin	– 0: Low: Accu0 counts if V < 4, Accu1 if V < 8, Accu2 if V < 16 – 1: LowMid: Accu0 counts if V < 32, Accu1 if V < 64, Accu2 if V < 128 – 2: UpMid: Accu0 counts if V ≥ 128, Accu1 if V ≥ 192, Accu2 if V ≥ 224 – 3: Up: Accu0 counts if V ≥ 240, Accu1 if V ≥ 248, Accu2 if V ≥ 252	Counts the amount of components V.
CROPEN (1 bit)		– 1: Statistics extracted from sub-rectangle – 0: Statistics extracted from whole frame	-
VSTART (12 bits)		-	Upper position of rectangle, from which statistics are extracted.
HSTART (12 bits)		-	Left position of the rectangle, from which statistics are extracted.
VSIZE (12 bits)		-	Height of the sub-rectangle, from which statistics are extracted.
HSIZE (12 bits)		-	Width of the sub-rectangle, from which statistics are extracted.
ACCU (24 bits)		-	Result of the 32-bit accumulation, re-sampled at frame boundary, and divided by 256 to fit the ACCU width of 24 bits.

Note: If HSIZE or VSIZE = 0 when the CROPEN bit is set, the block internally forces the "0" value at the maximum for the corresponding bit field (i.e. 0xFFE).

Table 397. Statistics extraction: collected data vs. modes

Mode	Average		Bin	
SRC	RGB	Raw	RGB	Raw
R, G, B up	012 = R / G / B	012 = RGB / 0 / 0	012 = R / G / B	012 = RGB / 0 / 0
L up	012 = R / 4 + G / 2 + B / 4	012 = RGB + 0 + 0	012 = R / 4 + G / 2 + B / 4	012 = RGB + 0 + 0
R, G, B down	012 = R / G / B	-	012 = R / G / B	-
L down	012 = R / 4 + G / 2 + B / 4	-	012 = R / 4 + G / 2 + B / 4	-

Configuration example for exposure control

For exposure control and white balance calibration, user would typically extract the average values of the luminance, and of the R, G, B. It can be done in a single frame capture campaign, with the following settings:

- ENABLE = 1 (enabled)
- MODE = 0 (average mode)
- SRC = 0 for red, 1 for G, 2 for B (upper values, before processing)
- BINS = 0 (all pixel values)
- CROPEN = 1 (1 below: to select a 300 x 220 close to top-left corner: 10,10)
- VSTART = 10
- HSTART = 10
- VSIZE= 220
- HSIZE= 300

After one frame (after a VSync), the Accu0, Accu1, Accu2 counters contain, respectively, the average red, green, blue accumulated across the 300 x 220 rectangle.

The value to expect from the Accu0, Accu1, Accu2 can be computed according to the following hypotheses:

- Amount of pixels: 300 x 220 (or full frame if statistics has CROPEN = 0)
- Amount of components:
  - Raw Bayer-red: 1/4, as one pixel every four in raw Bayer has a red component.
  - Raw Bayer-green: 1/2, as one pixel every two in raw Bayer has a green component.
  - Raw Bayer-blue: 1/4, as one pixel every four in raw Bayer has a blue component.
  - RGB: 1, as all pixels contain R, G, B components.
- Average value on 8 bits: 128 typically for a mid-gray picture (RGB: 128, 128, 128).
- Re-sampling: 1/256 (re-sampling accumulator to ACCU at end of frame).

The expected values in the accumulators are:

- Accu0 (red): 300 x 220 / 4 x 128 / 256 = 8250
- Accu1 (green): 300 x 220 / 2 x 128 / 256 = 16500
- Accu2 (blue): 300 x 220 / 4 x 128 / 256 = 8250

Configuration example for contrast enhancement

For contrast enhancement, the amount of pixels with a luminance inside a bin is typically extracted. As statistics are extracted 3 by 3, and as 12 bins can be accumulated, data are extracted during four consecutive frames, with three different bins for each frame.

The typical configuration is the following:

- ENABLE = 1 (enabled)
- MODE = 1 (bin mode)
- SRC = 3 (upper value, luminance selected)
- BINS = 0 (then, at next frame 1, then 2, then 3)
- CROPEN = 0 (no stat sub-rectangle)

After one frame (after a VSync), Accu0, Accu1, Accu2 counters contain the amount of pixels fitting bins <4, <8, <16, respectively.

The value to expect from the Accu0, Accu1, Accu2 can be computed by multiplying the following:

- Amount of pixels: 640 x 480 (assuming a full frame resolution of VGA).
- Amount of components: 1 (all pixels contribute to the luminance, even in raw Bayer).
- All pixels contribute to the statistics, both in raw Bayer and in RGB. In raw Bayer, a fake statistics is extracted, assuming a RGB-balanced pixel (Gray-world algorithm).
- Average value for bins: 256 (as in Bin mode, 256 is added for each fitting presence).
- Re-sampling: 1/256 (at end of frame)

The resulting value in accumulators are:

- Accu0 (bin < 4): 640 x 480 x 1 x 256 / 256 x 1% = 3072 (assuming 1% of pixels have lum < 4)
- Accu1 (bin < 8): 640 x 480 x 1 x 256 / 256 x 3% = 9000 (assuming 3% of pixels have lum < 8)
- Accu2 (bin < 16): 640 x 480 x 1 x 256 / 256 x 8% = 24000 (assuming 8% of pixels have lum < 16)

At the following frame, with BINS = 1, the statistics for luminance <32, <64, <128 are extracted, providing figures like 72k (25%), 144k (50%), and 210k (75%).

Such values indicate a median luminance ~64, as 50% of the pixels have a luminance <64.

42.6.12 Histogram

This module extracts from the ongoing frame histograms as statistic informations, and dumps them to external memory via the AXI.

The histograms can be computed from various location of the ISP pipe (before or after some key processing), computed on the full frame or across a grid of region, and one or more pixel components.

Functionality

- Configurable location of the sampled region(s) from which the histogram is extracted. It allows to focus the histogram on specific region of interest.
- Configurable amount of regions, shaped in a grid, with independent amount (vertically or horizontally), that can be 1 to 16: 1 to 4 histograms are extracted per region.
- Configurable bins per histogram: each histogram can contain 4, 16, 64, or 256 bins.
- Configurable dynamic of the input pixels. It is possible to concentrate on the dark pixels (to tune the black level), or in the opposite, to concentrate on the light pixels and the full dynamic (to tune the contrast). When the input pixel dynamic extends the bin dynamic, the upper bin collects the presence of all the too high pixel values.
- Configurable decimation (vertical and horizontal) of the extracted pixels, to tune the total sampled pixels and the local sampling rate.
- Configurable component selection: one or four among the available RGrBGb or RGBL.
- An interrupt is generated when the capture of the histogram statistics across the AXI has been completed. Two other interrupts warn that the software configuration is inconsistent or the memory bandwidth has not been high enough.

Software configuration

The software configuration register of the histogram (DCMIPP_P1HSCR) is shadowed, and can be updated from frame to frame.

Table 398. DCMIPP_P1HSCR, DCMIPP_P1HSSR, DCMIPP_P1HSMAR1, DCMIPP_P1HSMAR2, and DCMIPP_P1HSSMAR bit function

Bit ID	Function
EN	Activates the histogram extraction and write-out on the AXI to the external memory.
SRC	Selects the stage in the ISP pipe, from which the histograms are built. – Can be early and in raw Bayer pixel format, before any pixel processing after the decimation, or just before the exposure or the demosaicing. – Can be later, after the demosaicing and thus in RGB pixel format, before the ColorConv, or before the Contrast, or after the ISP processing. When a source is selected after the demosaicing, a RGB format is considered by the implementation.
HREG	Horizontal amount of regions per frame: 1 to 16 regions.
VREG	Vertical amount of regions per frame: 1 to 16 regions.
HDEC	Decimation in H of the input pixels, before extracting the histograms. A decimation factor of 1, 2, 4, 8, 16x can be applied. It makes possible to reduce the total amount of pixels to sample, or the pixel rate to sample.
VDEC	Decimation in V of the input pixels, before extracting the histograms. A decimation factor of 1, 2, 4, 8, 16x can be applied.
BIN	Bin amount per histogram, can be 4, 16, 64, or 256 bins. The bins are linearly spaced (they have the same width).
DYN	Selects the dynamic of the input components. The histogram can concentrate on the dark, the semi-dark, the semi-light, or the light value. The latter configuration (the recommended setting) balances the histogram across the whole dynamic range of the input sample.

Table 398. DCMIPP_P1HSCR, DCMIPP_P1HSSR, DCMIPP_P1HSMAR1, DCMIPP_P1HSMAR2, and DCMIPP_P1HSSMAR bit function (continued)

Bit ID	Function
COMP	Selects which components are extracted to build a histogram: 1 or 4 components are selected, and 1 or 4 histograms built. Depending on the input source, the four collected components are: – if rawBayer: R, Gr, B, Gb – if RGB: R, G, B, – if YUV: V, Y, U, V/4 + Y/2 + U/4 – if monochrome: 128, Y, 128, 128/4 + Y/2 + 128/4
HSTART VSTART HSIZE VSIZE	Location and size of the top-left region. The regions have all the same size. HSIZE and VSIZE must be multiple of their respective decimations.
MA	Memory address where the histograms are stored in the system memory. It is a multiple of 16 (the four LSBs must be 0). The address is configured in DCMIPP_P1HSMAR for the next frame. The last dumped address is in DCMIPP_P1HSSMAR register.

The histogram module is an open toolbox, that must be configured consistently. An inconsistent configuration leads to unpredictable and non-supported behaviors. The constraints are the following:

- Local storage capability:
  - The histogram module embeds a storage for 320 bins, shared for a line of region.
  - Depending on the amount of regions per line (1 to 16), and the amount of components to extract in parallel (1 or 4), a maximal amount of bins is available per histogram. The BIN field must select an amount not larger than the available one. In practice: #Region/line x #Component x #Bins ≤ 320.

  - For instance, with HREG = 4 (for a 5x5 grid, and thus five regions per line) and COMP = 4 (for four components extracted in parallel in four histograms), we have 320 / 5 / 4 = 16 bins maximum per histogram. BIN must be configured with 0 (four bins per histogram) or 1 (16 bins per histogram).
- Global decimation, versus the dynamic of each bin
  - Each bin is implemented with a 16-bit counter, and can thus accumulate a maximum of 64 k presences. The counter saturates afterwards at 65535, but trying precision of the stats.
  - It is recommended to configure the decimation in H and V, so that the amount of pixels extracted per region is smaller than 64 k (best) or not too high.
  - For instance, for a 5 Mpixel picture and one big region spanning the full frame, decimate the pixels by a ratio 5 M / 64 k = 76x. It can be configured with a 16x decimation in H, and an 8x decimation in V.
  - Note that when extracting a histogram from a raw Bayer source, the components, are already subsampled by the raw Bayer grid and its R, Gr, B, Gb sequence: only 25% of the pixels have a red component for instance.
- Decimation in H, versus the storage access rate:
  - The histogram extraction requires two clock cycles (of clk_proc) to store one component.
  - A minimal decimation in H is mandatory to provide the required free cycles to update the bins.
  - For instance, if COMP = 0 (or 1, 2, or 3), one component is extracted per pixel. One can thus extract one data every two pixels:
    > For an RGB pixel, it means a decimation by a factor 2x or larger, and thus configure HDEC = 1.
    > For a raw Bayer pixel, the interleaved R vs. Gr or B vs. Gb makes that any R, Gr, Gb, B component comes only once every two pixels, so that a decimation in H is not mandatory.
  - For instance, if COMP = 4, four components are extracted per pixel.
    > An RGB source requires a decimation of 8x or larger (HDEC = 3).
    > All the R, Gr, B, Gb components of a raw Bayer source are sampled, but one per cycle: the user must configure a decimation on H of a factor 2x or larger.
- Decimation in V, for the Line-of-region to line-of-region spacing:
  - After collecting the histograms across a full line of region, the histogram module exports the completed histograms via the AXI to the external memory, and reset them. For these tasks, 2 x 320 cycles must be provided after each line of region.
  - It is mandatory to configure a decimation in V, that frees 2 x 320 cycles. The decimation factor depends on the picture width. For a width of 640 pixels or more, a decimation in V of a factor 2x (VDEC = 1) is enough.
- All the regions from which the histograms are extracted must be defined (with VSTART, HSTART, VSIZE, HSIZE) to be fully within the frame. VSIZE and HSIZE define the size of a single region, that can be replicated many times in function of VREG and HREG.
  - HSTART and VSTART are not required to be multiple of the decimation, but must be even for raw Bayer pixels.

Note: When the source is raw Bayer, the decimation preserves the integrity of the pattern: in either H or V, a decimation by a factor 2x samples two pixels every four. A decimation by a factor

4x samples two pixels every eight, and so on. When no decimation is active, all pixels are sampled.

- When a grid of regions is configured, histograms for each region are computed and written to memory in sequence, starting at the location specified by the DCMIPP_P1HSMAR1 register. Regions within the grid are processed horizontally left to right and vertically top to bottom. Histogram bins are stored from lower to higher indices, and for each color component selected in DCMIPP_P1HSCR.COMP.

An interrupt is generated when the capture of the statistics for this frame has been completed on the AXI. The interrupt is the STFRAMEF located in both the DCMIPP_P1SR and the DCMIPP_CMSR2 registers.

Some software inconsistencies can be detected: if so, the interrupt STBCF is triggered in both the DCMIPP_P1SR and DCMIPP_CMSR2 registers. The main detected inconsistencies are for instance (such as usage of reserved/ parameters, too much temporary storage is used, size is not a multiple of the decimation).

In case of the AXI bandwidth has not been sufficient to store the statistics, an interrupt is generated: STOVRF in both the DCMIPP_P1SR and DCMIPP_CMSR2 registers.

In the last two cases, STBCF or STOVRF, the capture of any statistics at this frame is immediately aborted.

Configuration example 1

For a 5M (2560x2048) pixels sensor, full frame extraction across 5x5 regions of the raw Bayer R, Gr, Gb, B components in parallel, sampled at the source of the pipe before any processing:

- Local storage constraint: 320 bins, stored on five regions and four components, resulting in 320 / 5 / 4 = up to 16 bins/histogram.
- Global decimation constraint: because of the raw Bayer pattern, each component is present only 25% of the time, thus 1.25 M red components per frame (same for Gr, Gb, B). The frame is divided in 25 regions, hence 1.25 M / 25 = 51 k samples per region (for each component). It is lower than the recommended maximum (64 k), hence no need for decimation.
- Decimation in H: all the four RGrBGb components are sampled. As in raw Bayer, one component is sampled per pixel. To allow two cycles per component, a minimum decimation by 2x in H is needed.
- Decimation in V: the frame is 2560 pixels wide, more than the needed 640 free cycles between two lines or region. One single line between two lines of regions is needed, hence a decimation in V by 2x.

It results in the following configuration:

HDEC = 1	Decimation by 2x in H
VDEC = 1	Decimation by 2x in V
HREG = 4	To have five regions per line on the frame
VREG = 4	To have five regions vertically on the frame
HSTART = 0	Start at the left side of the frame
VSTART = 0	Start at the top of the frame
HSIZE = 128	2560 pixels / 2 (decimation factor) / 2 (decimation in raw Bayer) / 5 (region)

VSIZE = 100 2048 pixels / 2 (decimation factor) / 2 (decimation in raw Bayer) / 5 (region)

SRC = 0 Samples at the input of the ISP, after the BadPixelRemoval

COMP = 4 To build four histograms, one from each RGGB component

DYN = 0 Take the MSB, to sample the full pixels dynamics

BIN = 1 For 16 bins per histogram

EN = 1 To start the histogram extraction and dump

Configuration example 2

For a 5M (2560x2048) pixels sensor, full frame extraction with a single large region of the single luminance component (extracted from a RGB pixel format), sampled after demosaicing and before the ColorConv stage:

- Local storage constraint: 320 bins, stored on one region and one component, resulting in 320 / 1 / 1 = up to 320 bins/histogram, rounded to the configurable value of 256.
- Global decimation constraint: the frame is 5 Mpixel, versus a recommended maximum of 64k samples per region. It results in 5 M / 64 k = 76x decimation, shared best among 16x in H, and 8x in V.
- Decimation in H: one components per pixel is sampled, resulting in a decimation by a factor 2x minimum. The above decimation in H by 16x sets a larger constraint.
- Decimation in V: with 2560 pixels wide, one line is needed between two lines of regions, thus a decimation in V by 2x minimum. The above decimation in V by 8x is a larger constraint.

It results in the following configuration:

HDEC = 4 Decimation by 16x in H

VDEC = 3 Decimation by 8x in V

HREG = 0 To have a single region, one per line on the frame

VREG = 0 To have a single region, one vertically on the frame

HSTART = 0 Start at the left side of the frame

VSTART = 0 Start at the top of the frame

HSIZE = 160 2560 pixels / 16 (decimation factor) / 1 (region)

VSIZE = 256 2048 pixels / 8 (decimation factor) / 1 (region)

SRC = 2 Samples at the output of the demosaicing

COMP = 3 To sample the computed luminance

DYN = 0 Take the MSB, to sample the full pixels dynamics

BIN = 3 For 256 bins per histogram

EN = 1 To start the histogram extraction and dump

Simplified configuration and constraints

A simplified set of constraints can be defined by setting a conservative configuration for the decimation, that fits any case, at the expense of fewer samples and less precision in the histograms.

A decimation by 16 horizontally and 8 vertically fits all the constraints, whatever the resolution or sampling factor.

Assume a Width and Height of the input frame:

HDEC =	4	RGB source	Start/size counts per 16 pixels
	3	Raw Bayer source	
VDEC =	3	RGB source	
	2	Raw Bayer source	
HSTART = 0		Left side, always full frame	
VSTART = 0		Start at the top of the frame	
HSIZE = Width / 16 (decimation) / R (regions)			
VSIZE = Height / 16 (decimation) / R (regions)			
SRC = any		Samples at the input of the ISP, after the BadPixelRemoval	
HREG = any		Any amount, 1 to 16 of regions per line	
VREG = any		Any amount, 1 to 16 of regions vertically per frame	
COMP = any		Any amount, 1 to 4 components	
DYN = 0		Take the MSB, to sample the full pixels dynamics	
BIN = compute		In function of 320 bins / R (regions) / C (components)	
EN = 1		To start the histogram extraction and dump	

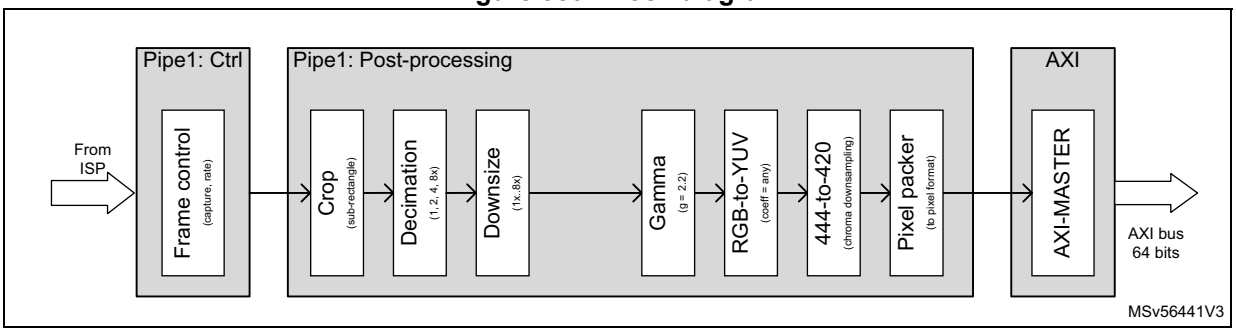
42.7 Pipe1 (post-processing part)

42.7.1 Overview

The post-processing Pipe1 (whose diagram is shown in *Figure 350*) is a pixel pipeline in which post-processing functions are integrated to relieve the software from very specific operations (time and memory consuming), like:

- Crop, to keep only the part of the picture required by the targeted application
- Decimation, as a cheap preprocessing before the downsize
- Downsize, to reduce the picture resolution (size reduction)
- Gamma correction before sending the data to the memory for further displaying operations
- YUV conversion with possible down-sampling option, to optimize the memory size needed to store the corresponding image or to match a YUV420 video encoder.

Figure 350. Block diagram



The applicative Pipe1 is the most featured pixel pipeline, even if the image processing functions can be shared with Pipe2 (refer to *Section 42.8.2*).

Pipe1 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 42.13.2*).

It is also possible to write into the shadow registers during the processing of a frame. The values are loaded into the physical register based on the following start of frame event.

42.7.2 Pixel 2D cropping

The pixel 2D cropping aims at extracting and forwarding a rectangle of pixels.

The cropping assigns an X and Y location to its input pixels, and selects a configured rectangle among the pixels, and forward the pixels of this rectangle farther in the pixel pipe. The pixels around the selected rectangle are discarded.

Functionality

- Counts lines thanks to HSync (if from parallel interface) or packets (if from CSI-2)
- Counts pixels inside a line
- Positive cropping (forwards a selected rectangle of pixels in the pipe)
- Cropping with 0 value for HSIZE[11:0] or VSIZE[11:0] forces the internal crop to get the maximum value on the parameter where the software sets to the 0 value

Software configuration

Table 399. DCMIPP_PxCRSTR and DCMIPP_PxCRSZR bit function

Bit ID	Values	Comments
ENABLE	– 1: Cropping active – 0: All data transmitted	-
HSTART, VSTART	0 to 4094	Top-left corner of the cropped rectangle.
HSIZE, VSIZE	0 to 4094	Width and height of the cropped rectangle. Can be larger than the received picture to grab everything.

If the cropped window is larger than the received window, only the received pixels are forwarded (no pixels are inserted to match the window size).

If the software sets 0 in one or both horizontal/vertical size bit fields (HSIZE, VSIZE), the crop forces internally the value to be at the maximum on the corresponding bit field.

Configuration example

To select a small rectangle (size 100 x 50), whose upper pixel is located at position (30, 10), provide the following setting:

- ENABLE = 1 to crop a sub-rectangle
- HSTART, VSTART = (30, 10) to start cropping at x = 30, y = 10 of the received picture
- HSIZE, VSIZE = (100, 50)

The most bottom-right selected pixels have a position (129, 59) from the received picture.

The frame forwarded is handled downstream in the pipe as a 100 x 50 frame.

42.7.3 Decimation pre-downsize

The decimation (prior to downsize) makes it possible to cheaply downsize a frame by a factor 2, 4, or 8x, in both horizontal and vertical directions.

The decimation is partially redundant with the downsize, that brings a higher quality (the decimation discards the data of the dropped pixels, while the downsize averages all pixels and does not lose information).

The decimation complements the downsize (limited to ratio of 8 x 8), to have a combined downsizing ratio of 64 x 64. It is recommended to have the largest ratio implemented in the higher quality downsize, and have needed the decimation complement to that. For instance, for a ratio 30x, is it recommended to have a ratio 4x in decimation, and a ratio 30 / 4 = 7.5 in the downsize.

Features

- Forwards one item (typically pixel) every 1, 2, 4, 8 pixels, independently vertical and horizontal.

Software configuration

Table 400. DCMIPP_PxDCCR bit function

Bit ID	Function	Value	Comments
HDEC[1:0]	Ratio (2 bits)	0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8
VDEC[1:0]		0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8

42.7.4 Downsize

This module implements a downsize based on box filtering, which allows ratios from 1x to 8x, independently, in both X and Y directions.

The box filter has properties close to a bilinear filter for small ratios (from 1x to 2x), but provides the advantage of avoiding some Moiré effect for higher ratios (from 2x to 8x) thanks to using all the samples.

Functionality

- Features:
  - Integer + decimal ratio, different vertically and horizontally.
  - Ratio: 1.0x to 8x
  - Limited to a maximum output buffer width of 2688 pixels due to the needed linebuffer.
- Operations:
  - Increments defined with 3.13 precision (<1 pixel error after maximum width of 4094)
  - Internal computations for filter defined with 3.3 precision
  - Accumulates skipped pixels to allow ratio up to 8x with saturation on 11 bits
  - Weighting of shared pixels, to allow good quality at ratio close to 1
  - Independent H filter, then V filter
  - Init phase: left of left-most pixel is aligned on input pixel (see below Pi0, Po0)

Software configuration

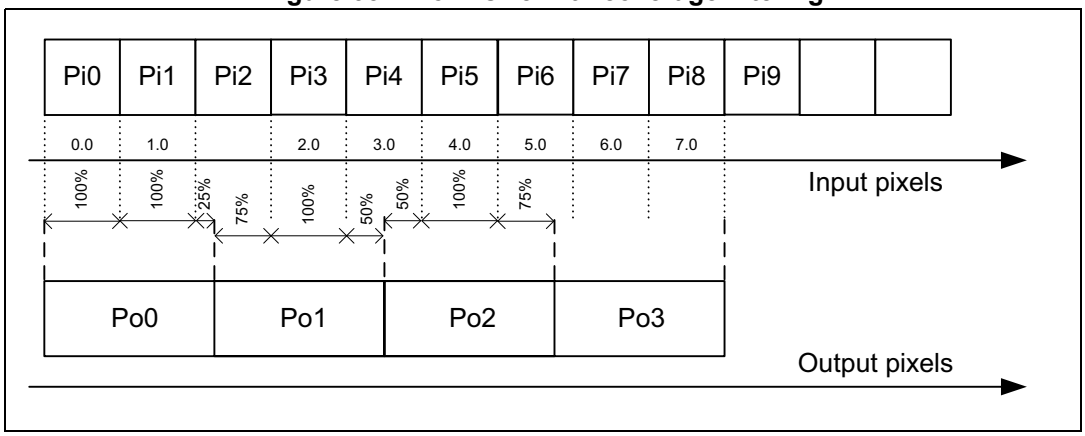
Table 401. DCMIPP_PxDSCR, DCMIPP_PxDSRTIOR, DCMIPP_PxDSSZR

Bit ID	Function	Values	Comments
ENABLE	-	– 1: Downsize active – 0: Transmission without downsize	-
HRATIO, VRATIO	Downsize ratio (respectively horizontally and vertically)	16-bit fields, used as unsigned 3 integer 13 decimal. Valid values are 8192 (= 1x) to 65535 (= 7.999x).	Key parameters that create output pixels. The optimal is computed as xRATIO = Floor (8192 * xRatioFloatingPoint), maximum value 65535 (x = H or V).
HDIV, VDIV	Inverted ratio, must be consistent with Ratio	10-bit fields, used as unsigned 10 decimal. Valid values are 128 (= inv8x) to 1023 (= inv1x).	A side parameter, impacts only the luminance of the output pixels. The optimal in function of xRATIO is xDIV = Floor((1024 * 8192 -1) / xRATIO) (x = H or V).
HSIZE, VSIZE	Final size after the downsize	-	The downsize filter clips pixels if the post-downsize size is not correct. The HSIZE and VSIZE are partially redundant vs. the cropping size divided by the downsize ratio. If VSIZE = 0 or HSIZE = 0, the filter is disabled in that dimension.

Figure 351 shows the principle of the computations:

- Po0 = (1.00 x Pi0 + 1.00 x Pi1 + 0.25 Pi2) / 2.25
- Po1 = (0.75 x Pi2 + 1.00 x Pi3 + 0.50 Pi4) / 2.25
- Po2 = (0.50 x Pi4 + 1.00 x Pi5 + 0.75 Pi6) / 2.25

Figure 351. Downsize with coverage filtering



Configuration example

- ENABLE = 1 to activate the downsize
- Downsize ratio 1x: HRATIO = VRATIO = 8192, HDIV = VDIV = 1023
- Downsize ratio 2x: HRATIO = VRATIO = 16384, HDIV = VDIV = 512
- Downsize ratio 3x: HRATIO = VRATIO = 24576, HDIV = VDIV = 341
- Downsize ratio 8x: HRATIO = VRATIO = 65535, HDIV = VDIV = 128

42.7.5 Gamma conversion

This feature implements a gamma compression on each R, G, B component, using a static gamma exponent 2.2, to store gamma-compressed pictures, as defined in the sRGB standard.

Features

- The gamma is implemented as a 7-segment linear curve.
- It crosses the following way-points, in logarithm spacing:
  - (0,0)
  - (4,39)
  - (8,53)
  - (16,72)
  - (32,99)
  - (64,136)
  - (128,186)
  - (256,256)

Software configuration

ENABLE = 1 in the DCMIPP_PxGMCR registers activate the color conversion.

42.7.6 YUV conversion

This block is a replication of the one described in *Section 42.6.9*, used to convert RGB in YUV color format (basically, since the conversion YUV to RGB is assumed to be performed by the color conversion block into the image processing main function). Bit fields are the same of the color conversion module, only the register name and the memory mapping are different. The register functions are the same, the register names are DCMIPP_P1YUVxxx instead of DCMIPP_P1CCxx.

42.7.7 Chroma down-sampling

The Chroma down-sampling module reduces the chroma resolution from its arriving YUV444 format to the potential YUV420 output, and sends them to a different channel.

Features

- Downsizes in H first, if YUV420, with (0.5 + 0.5) filter
- Downsizes in V then, if YUV420, with (0.5 + 0.5) filter
- Memory: embeds one half-line buffer for each U and V

Software configuration

Configuration is indirectly taken from the pixel packing output format, located in the DCMIPP_PxPPCR register.

- The Chroma down-sampling is active when the output pixel format is YUV422-1, YUV422-2, YUV420-2, YUV420-3.

Output

- One to three channels: RGB/Y, UV (or U) and V.

42.7.8 Pixel packing

This module works on the dump pipe (Pipe0) and pixel pipes (Pipe1 .. N), setting the arriving pixels in memory words.

Features

- Single address (single buffer base-address): if a double or triple buffer is needed, software must update the single base-address at each frame.
- Flexible pitch (line stride): width up to 4094 pixels, 32 bpp, potentially interleaved.
- Pixel format: all usual formats, as indicated in *Section 42.3.6* (Pipe1 allows semi-planar and full planar buffer, Pipe2 coplanar buffer only).
- Swap R-vs-B: swap the red vs. the blue components of the output pixel.
- Padding: MSB vs. LSB padding of 10/12/14 bit onto 16 bit. (such formats are supported only on the dump pipe).
- Header enable: allows to dump (or not) the CSI packet headers, to allow software an easier extraction of data in case of Interleaved CSI packets in the captured buffer.
- Multi-line event: event generated (for DMA and INT) at each multiple of sent lines.

Note: Some features (for example padding) are available only on the dump pipe, while others (like swap R-vs-B) are available only on the pixel pipes.

Table 402. DCMIPP_PxPPCR bit function

Bit ID	Function	Value	Comments
M0A, M1A, M2A	Base address of pixel buffer in memory, 32-bit wide, aligned on 16 bytes	-	The line alignment of the pixel buffers in memory is assumed on 16 bytes. Therefore the buffer base address (i.e. M0A, M1A, M2A) must be a multiple of 16 bytes, and similarly for line PITCH.
PITCH	Address jump from line to line, up to 32 Kbyte - 1.	-	Aligned on 16 bytes.
SWAPRB	Swaps R-vs-B components (and U-vs-V, same operation if Y mapped on G) available on pixel pipe and not on dump pipe.	-	-
HEADEREN	-	– 1: Dumps headers	Feature present only in dump pipe.

Table 402. DCMIPP_PxPPCR bit function (continued)

Bit ID	Function	Value	Comments
PAD	MSB vs. LSB padding of the raw Bayer and monochrome to 16 bpp.	– 0 (align on LSB): backward compatibility with the former DCMI IP – 1 (align on MSB): easier use by software or GPU	Applies only on the dump pipe and for the 10/12/14 (not 8)-bit pixels (the pixel pipe convert pixels into 8 bpc, and padding is therefore not relevant).
FORMAT	Output pixel format	-	– Pipe0: not defined as default input format used for dump. It allows unique formats ByteData, ByteHeader and Mono/raw Bayer 8/10/12/14. – Pipe1: adds semi- and multi-planar formats: YUV 422-2, 420-2, 420-3 if Alpha or X is dumped, they are forced at 255). – Pipe2: supports coplanar pixel formats xRGB8888 (=xYUV444), RGB888 (=YUV444), RGB565, YUV422-1 and Y8.
LINEMULT	Periodicity of the line (HSync) interrupt and event, as power of 2 of LINEMULT configuration (i.e. every 1, 2, ..,128 lines).	-	-

42.7.9 Overrun detection

This logic handles flow-control hazards: the DCMIPP is provided a continuous flow of data (via parallel or CSI2 interfaces) without any wait/hold capability, while downstream of DCMIPP, the access to memory can be temporarily stuck.

In some cases the internal FIFOs of the DCMIPP get full and pixels may be deleted. The overrun detection logic handles such hazard at best.

Features

- Software gets warned in case of an hazard:
  - Interrupt OVR is triggered at the first deletion of a pixel, repeated at every frame, as long as the overrun hazard is present.
  - It is recommended to skip / trash the impacted frame:
    > Overflowing pixels are deleted, so that when the access to memory resumes, the remaining of the line is discarded (not written out). The normal processing resumes at the start of the next line. Potentially, several full lines can get discarded.
- No Frame-to-Frame impact in case of an hazard:
  - VSync resets what is needed in the DCMIPP, so that any hazard on a previous frame has no impact at all on the pixels dumped in next frames.

Software configuration

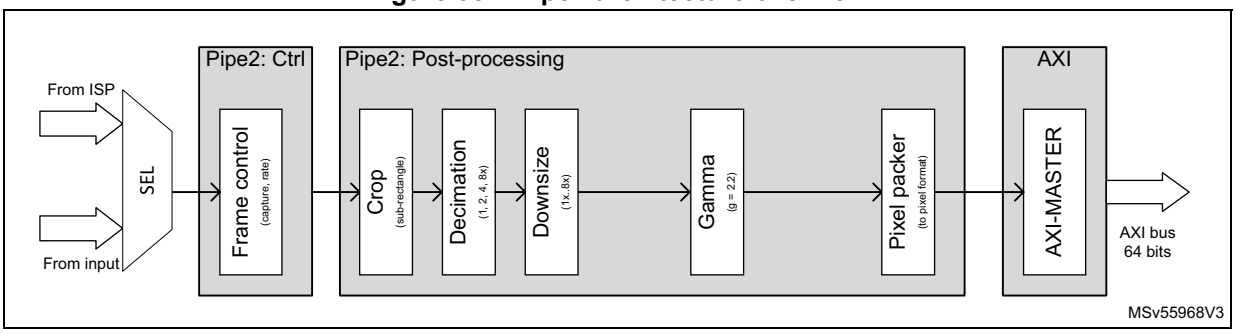
- Interrupt DCMIPP_PxSR.OVRF is warned in case of hazard.

42.8 Pipe2 (post-processing)

42.8.1 Overview

Pipe2, compared to Pipe1, is a lighter pixel pipeline offering no embedded image processing blocks. This pipe can work in standalone mode, or share the image processing functions with Pipe1 (refer to *Section 42.8.2*).

Figure 352. Pipe2 architecture overview



The main blocks (controller and post-processing general functions) of Pipe2 are almost the same as those of Pipe1. The flow selection and frame controller allow to configure Pipe2 behavior when it is used in standalone, meaning without image processing heritage from Pipe1. Refer to *Section 42.4.6* and *Section 42.4.7* for more details about the use cases and the configuration of the pipe.

The set of post-processing blocks is reduced compared to Pipe1, as neither the YUV conversion nor the down-sampling YUV444 to YUV420 are integrated in Pipe2. This pipe can have its own image crop area, its dedicated downsizing, and take benefit of the gamma correction, like Pipe1. Refer to *Section 42.7* and to its subsections for more details on how to use and configure by software these post-processing functions.

Software configuration

To use Pipe2 as a standalone pixel pipe, without any image processing functions shared with Pipe1, bit PIPEDIFF must be set in the DCMIPP_P1FSCR register. This bit has a meaning only when the flows can be differentiated, that is, when the camera connected and processed by the DCMIPP is a CSI2 camera interface.

In this non-shared mode, it is required that different flows are sent to Pipe1 and Pipe2, or a non-supported behavior can occur.

The flow to Pipe1 and the flow to Pipe2 may potentially be interleaved at packet level.

Pipe2 may be reconfigured time to time. Some events may be used to trigger the software reconfiguration routines (refer to *Section 42.13.2: Interrupts*).

It is possible also to write into the shadow registers within the processing of a frame. The values are loaded into the physical register based on the start of frame event.

42.8.2 Pipe1 and Pipe2 sharing image processing functions

Following the application, it may be interesting to share the image processing functions belonging to Pipe1 with Pipe2 when the dataflow is shared for both. One pipe may be used for display operation, and the other one used for statistics or software analysis. In this case PIPEDIFF bit of DCMIPP_P1FSCR register must be cleared.

If the CSI-2 module is connected to the DCMIPP, the VC DT and DTMODE of Pipe2 are the same as those configured into DCMIPP_P1FSCR register, making non relevant the values for these fields into the DCMIPP_P2FSCR register. However, bits CPTREQ and CPTMODE remain active into the DCMIPP_P2FCTCR register to control the flow inside Pipe2.

42.9 Application use cases

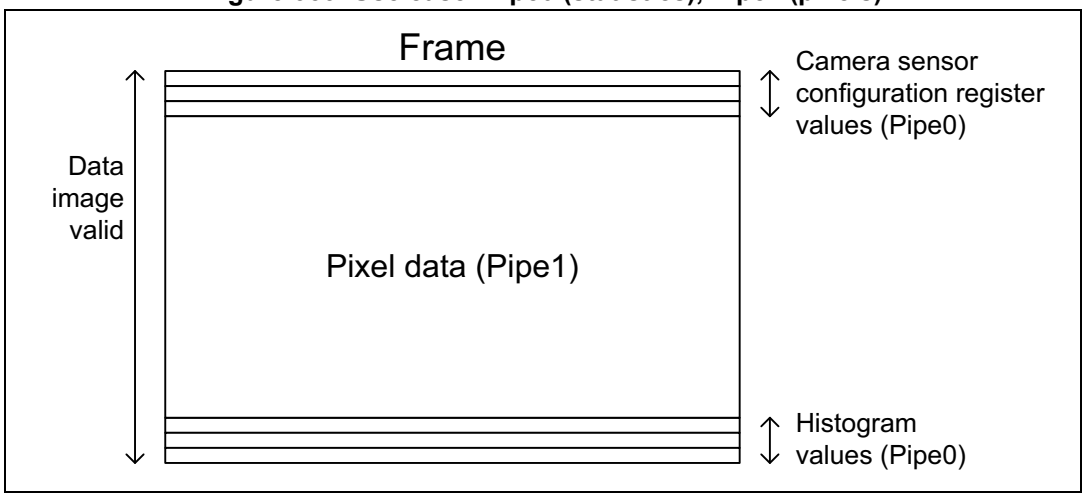
There are several application use cases that can fit the DCMIPP architecture, a few of them are described in this section.

42.9.1 Parallel interface camera sensor module

A parallel interface camera sensor can transfer images as well as statistics. It is easier to manage the software processing by splitting the statistics and the image data on different memory storages. Two pipes are used in such case, with Pipe0 extracting the statistics and Pipe1 dealing with the image data (whatever the data format supported by the DCMIPP). Both work in parallel on the same frame, frame by frame. Each output FIFO attached to the pipe conveys the data on the targeted memory for further processing.

The camera sensor module may use some lines of the frame to transfer statistics or sensor configuration data. These lines can be placed at the beginning and at the end of the frame (see *Figure 353*), even if there is no real standard.

Figure 353. Use case: Pipe0 (statistics), Pipe1 (pixels)



Software configuration

- Pipe0: select the crop area to extract statistics only
  - Configure HSTART[11:0] = 0 and VSTART[11:0] bits in the DCMIPP_P0SCSTR register from the line number corresponding to the first pixel data line (after first statistics line).
  - Configure HSIZE [11:0] (horizontal screen resolution in number of data 32-bit equivalent to the horizontal resolution of the screen in pixels) and VSIZE [11:0] to

the value corresponding to the offset line number from the first pixel data area. These bits are in the DCMIPP_P0SCSZR register.

– Set bit POSNEG of the DCMIPP_P0SCSZR register to reject data belonging to the rectangle defined, to keep the outside area of the rectangle, where the statistics are taken in this use case.

- Pipe1: Select the area to extract the statistics into DCMIPP_P1SRCR register, by configuring bit FIRSTLINEDEL[2:0] and LASTLINE[11:0] (it avoids to start the image processing on statistics data). Only the pixel data are forwarded, if needed, to the image processing blocks (for example demoisaicing, auto exposure, contrast enhancement) for further processing.

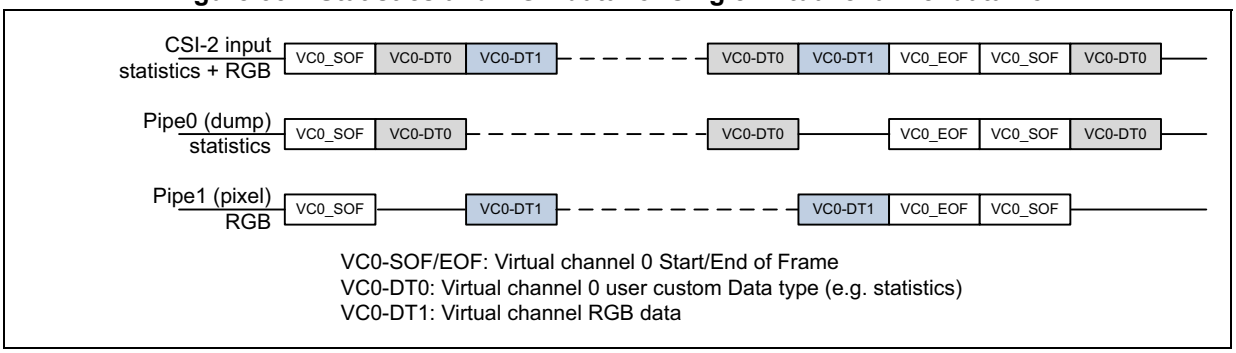
42.9.2 CSI2 camera sensor module

The CSI2 protocol allows, thanks to Datatypes, to make distinction between different data categories. The application may request to handle interlaced video as well as interleaved data flow like described below.

Single virtual channel involving two pipes (for example Pipe0 and Pipe1)

- Use case: the CSI provides two data flows in parallel, typically RGB (or raw Bayer) with its sideband statistics.
- Handling: the statistics are dumped by the dump pipe, while the pixels are sent to the pixel pipes.

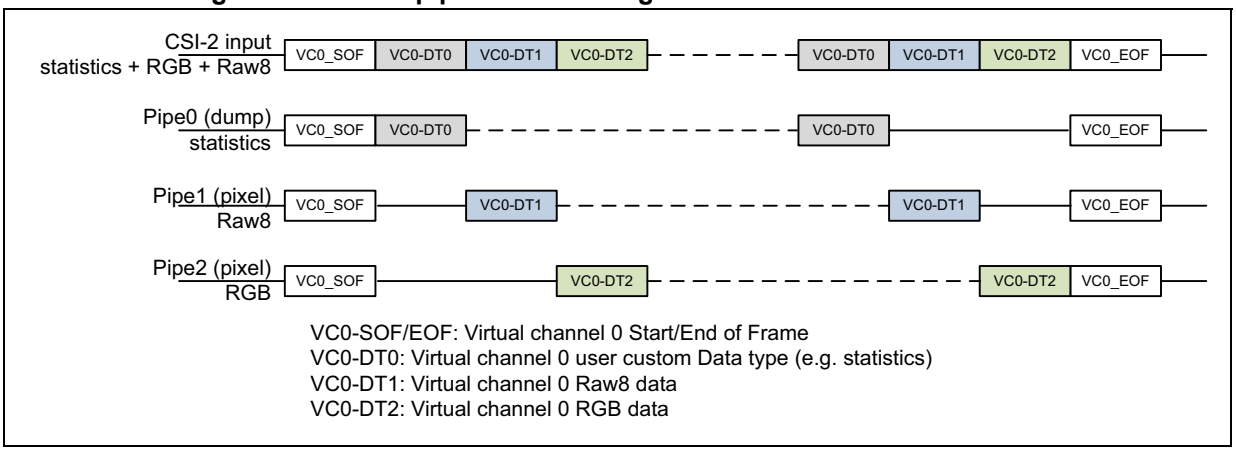
Figure 354. Statistics and RGB data for single virtual channel data flow



Single virtual channel involving three pipes (for example Pipe0, Pipe1 and Pipe2)

Another use case (*Figure 355*) employs different Datatypes in addition to the statistics within the CSI-2 flow based on a single virtual channel.

Figure 355. Three pipes within a single virtual channel in CSI2 mode



Interleaved packets (based on multiple virtual channels)

- Use case: the CSI provides two pixel flows in parallel, typically a JPG-compressed flow and an RGB888 or raw Bayer flow.
- Handling: the JPG is dumped by the dump pipe, while the pixels are sent to the pixel pipes.

Both Pipe1 and Pipe2 can process these pixels. The shared demosaicing mode can be used, hence both Pipe1 and Pipe2 benefit of the potential demosaicing conversion.

Figure 356. Handling interleaved packet

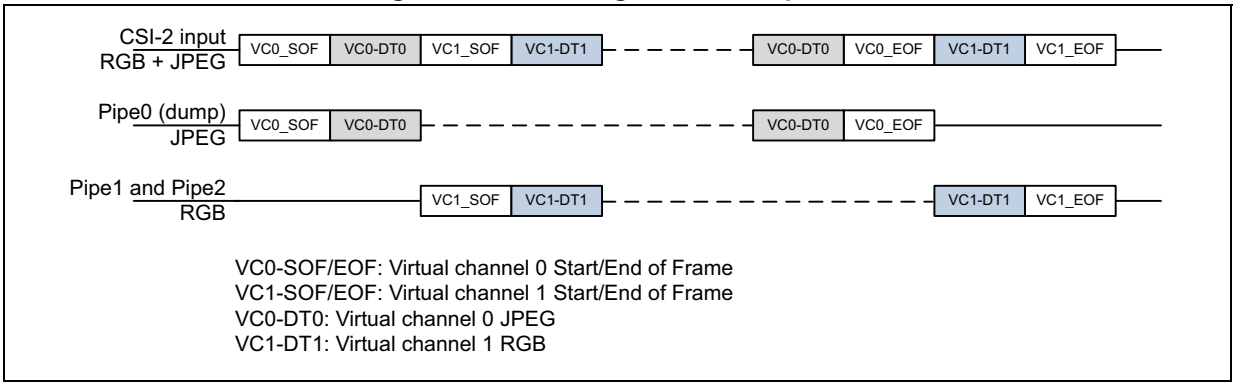
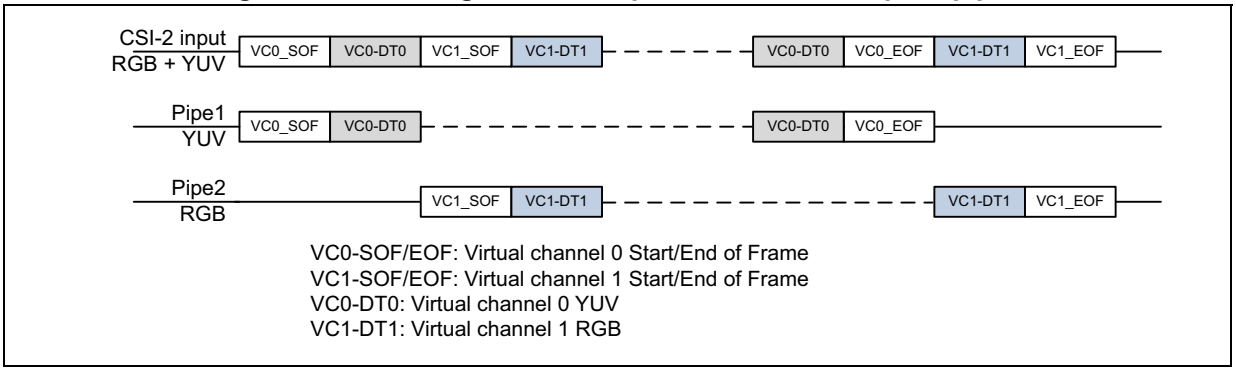


Figure 357. Handling interleaved packets across two pixel pipes



Interlaced video

- Use case: the CSI provides the odd and even frames of an interlaced video onto two different virtual channels (usually with a same data type).
- Handling: the flows of both the virtual channel of the odd field and the virtual channel of the even field are sent to the pixel pipes.
- Single pixel pipe may be configured if the image processing function is required by the application to handle both interlaced frame (refer to *Figure 358*). In such case, a pipe reconfiguration is needed frame by frame to change the virtual channel number and the address memory for the output buffer if the software distinctly keeps the odd and even frame buffers in their respective memory locations.
- Pipe1 and Pipe2 may be used if the application software does not need image processing functions (refer to *Figure 359*). Each pipe is configured with its own virtual channel and its own output buffer address in memory. The split of the odd and even frames is explicitly defined in a static way, avoiding any pipe reconfiguration between two frames.
- Interlaced video can be managed with a single virtual channel identifier, using a different data type to identify odd and even frames. In such case, the CSI2 bridge uses user Datatypes to distinguish between them.
- Limitation: the frames of the two virtual channels must be time-exclusive, by default the case for usual interlaced video sensors.

Figure 358. Interlaced video (based on single pixel pipe)

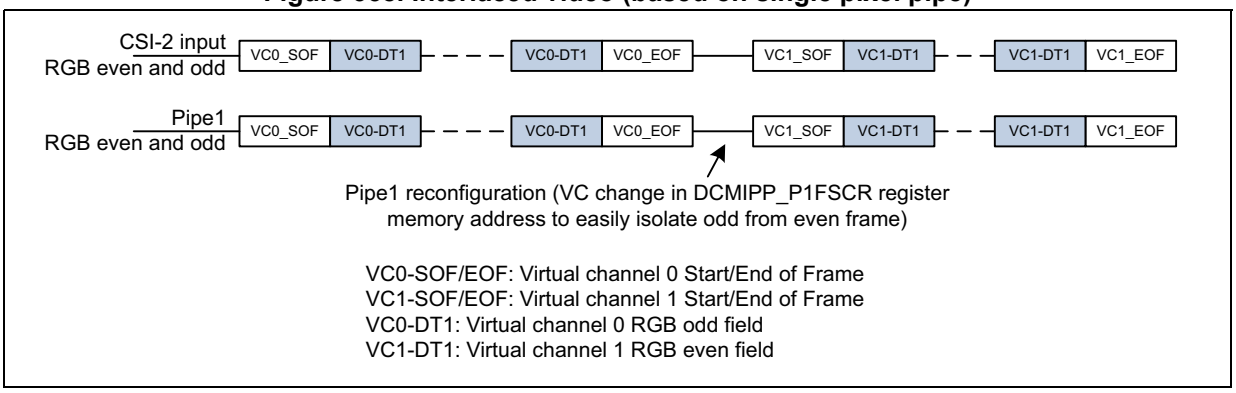
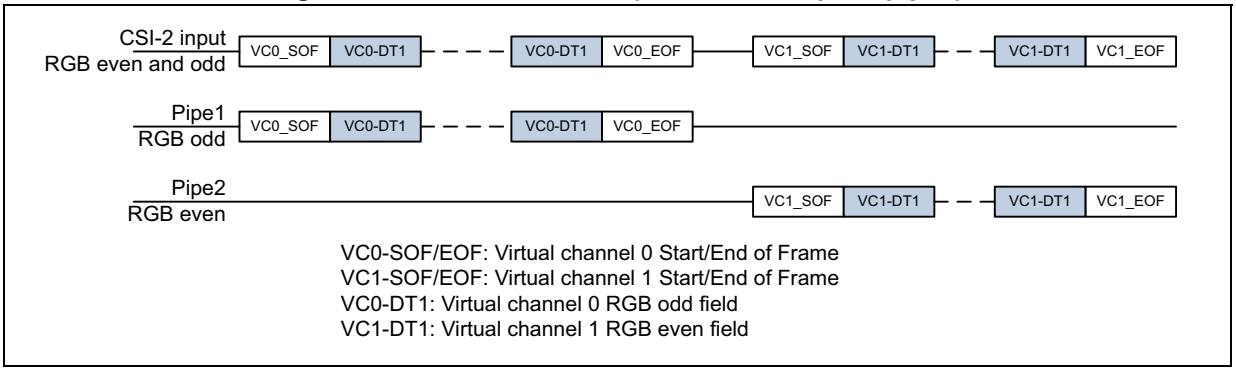


Figure 359. Interlaced video (based on two pixel pipes)



Software configuration

The following configuration must be set first into the static DCMIPP_PxFSCR and DCMIPP_CMCR registers:

- INSEL: Selects the input flow (parallel vs. CSI) in the DCMIPP_CMCR register
- DTMODE[1:0]: mode for CSI flow selection into that pipe:
  - 0: only flow VC/DTIDA is forwarded in the corresponding pipe.
  - 1: flows VC/DTIDA and/or VC/DTIDB are forwarded for Pipe0, and VC/DTIDA and VC/DTIDB for Pipe1.
  - 2: all flows, except the VCDTIDA or VCDTIDB, are forwarded in the pipe (this last setting, 3, is available only for Pipe0).
  - 3: all Datatypes from the selected virtual channel are forwarded in Pipe0. It is valid only for Pipe0 (the pixel formats that are not supported by Pixel pipes are suppressed).

  This bit field is not valid for Pipe2.
- DTIDA/DTIDB: Data type selection ID-A and ID-B for Pipe0 and Pipe1.
- DTIDA: Data type selection for the pixel Pipe2 when PIPEDIFF = 1.
- FDTFEN: Forces the value of the data type (value defined in FDTF[5:0]) to be processed by the pixel Pipe1 and Pipe2, based on the DTIDA/DTIDB detection in the flow selection block of the pipe. Not available for Pipe0.
- FDTF[5:0]: Data type value overwritten the one received by the CSI-2 physical node when equal to DTIDA/DTIDB. Not available for Pipe0.
- PIPEDIFF: Pipe2 to take same input as Pipe1 or differentiated:
  - 0: Pipe2 takes the same VC/DTIDx as well as the same FDTFEN and FDTF[5:0] configured into the DCMIPP_P1FSCR register.
  - 1: Pipe1 and Pipe2 take their own VC/DTID, FDTFEN and FDTF[5:0] values in the data flow.

After all the pipes have been configured, the processing can be activated:

- PIPEN: Set to 1 to start the frame control logic of the pipe. When reset at 0, it immediately resets the frame control logic, which results in immediately stopping an ongoing dump (CPTREQ = 0, CPTACT = 0), see *Figure 344*, *Figure 345*, *Figure 346*.
- CPTREQ: Set to 1 to activate the capture at the next start-of-frame in the DCMIPP_PxFCTCR register.

Note: To be cleanly sampled for the next frame, the VCIDA, VCIDB, DTIDA, and DTIDB fields must be updated after a VSync and after the first HSync.

Configuration example

The following list details five possible use cases, among others interlaced video and interleaved frames, and a 3D sensor (left + right eye):

- Default, with a parallel dump of statistics and processing of an image on both Pipe1 and Pipe2:
  - Image forwarded to the pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract a single flow.
    - Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the image flow
    - Pipe1 / PIPEDIFF = 0, to send the flow to both Pipe1 and Pipe2
  - Statistics are extracted and dumped on Pipe0 with the following config:
    - Pipe0 / DTMODE = 0, to extract a single flow.
    - Pipe0 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the statistics flow.
- Interlaced video, with the consecutive but exclusive odd and even (requiring the use of the image processing functions):
  - Interlaced video, if having differentiated VC and DT, is forwarded to the pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract the VC /DTIDA flow
    - Pipe1 / VC/DTIDA = vv_dddddd, with VC and DTIDA matching the odd or the even field by reprogramming between two frames
    - Pipe1 / PIPEDIFF = 0, to send the flow to both Pipe1 and Pipe2
    - Pipe1 / Pipe2 / CPTMODE = 1, Snapshot mode
    - Pipe1 / Pipe2 / CPTREQ = 1 alternatively for Pipe1 and Pipe2, frame by frame
  - Any other (different) data can be potentially dumped with the following settings:
    - Pipe0 / DTMODE = 2, to extract all but flows A and B
    - Pipe0 / VCDTIDA = Pipe1 / VC / DTIDA corresponding to the odd field
    - Pipe0 / VCDTIDB = Pipe1 / VC / DTIDB corresponding to the even field

The software needs to reprogram frame to frame the DTIDx fields of the DCMIPP_P1FSCR register to select the data to process into Pipe1 or Pipe2. The software sets CPTMODE bit to work in Snapshot mode for Pipe1 and Pipe2 in the DCMIPP_P1FCTCR and DCMIPP_P2FCTCR registers. Alternatively, bit CPTREQ is set frame by frame in one pipe then in the other one to allow the acquisition of data on the allocated pipe. The data type identifying the odd or even fields needs to by updated into the DCMIPP_P1FSCR register only, as Pipe2 shares the image processing block of Pipe1.

- Interlaced video, with consecutive but exclusive odd and even (bypassing the image processing functions):
  - Interlaced video, if having differentiated VC and/or DT, is forwarded to pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract the VC /DTIDA flow
    - Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching (for instance) the odd field data type
    - Pipe2 / DTMODE = 0, to extract the VC /DTIDA flow

> Pipe2 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching (for instance) the even field data type
> Pipe1 / PIPEDIFF = 1, to consider Pipe1 and Pipe2 as totally independent
> Pipe1 / Pipe2 / CPTMODE = 0, Continuous grab mode
> Pipe1 / Pipe2 / CPTREQ = 1 to capture data corresponding to the VCDTIDA attached to the corresponding pipe

– Any other different data can be potentially dumped with the following settings:
> Pipe0 / DTMODE = 2, to extract all but flows A and B
> Pipe0 / VCDTIDA = Pipe0 / VC / DTIDA corresponding to the odd field
> Pipe0 / VCDTIDB = Pipe0 / VC / DTIDB corresponding to the even field

In this case the interlaced video must be differentiated with the data type and not with the virtual channel (since there is a single VC by pipe), if the application wants to dump data other than the odd and even frames on Pipe0.

- Interleaved flows, with a parallel dump of an RGB picture and a JPEG-encoded flow:
  – The RGB picture is extracted and forwarded to pixel Pipe1 to be processed, with the following settings:
  > Pipe1 / DTMODE = 0, to extract one flow, the RGB picture
  > Pipe1 / VC /DTIDA = vv_dddddd, with VC and DTIDA matching the RGB picture data type
  > Pipe1 / PIPEDIFF = 1, to send the RGB on Pipe1 and use Pipe2 for other purposes

  – The JPEG-encoded stream is transmitted under the CSI user-defined format, is extracted and sent to the pipe to be dumped, thanks to the following:
  > Pipe0 / DTMODE = 0, to extract one flow, the JPEG-encoded flow
  > Pipe0 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the JPEG stream data type
  > Pipe0 / VCDTIDB = not used

  – Another pixel picture, like YUV, can be dumped using Pipe2, by overwriting fields:
  > Pipe2 / DTMODE = 0, to extract a second flow for the YUV picture
  > Pipe2 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the YUV picture data type

- 3D sensor, with a parallel transmission of two RGB or YUV pictures: one with data type DT1 for the left eye, and one with data type DT2 for the right eye: the DCMIPP can process each eye on a different pipe (for example left eye on Pipe1 and right eye on Pipe2). Note that, as Pipe1 and Pipe2 are differentiated, the demosaicing is assigned to Pipe1 only and is not available for Pipe2. The left and right eye flows are similar (raw Bayer or RGB), hence they both need to be RGB and not raw Bayer:
  – The left eye RGB picture is extracted and forwarded to the pixel Pipe1:
  > Pipe1 / DTMODE = 0, to extract one flow, the RGB picture
  > Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the left eye data type (DT1)
  > Pipe1 / PIPEDIFF = 1, to split and differentiate the flow between Pipe1 and Pipe2

  – The right eye RGB picture is extracted and forwarded to the pixel Pipe2:
  > Pipe2 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the right eye

data type (DT2)

  - Another flow can be dumped onto the dump pipe:
    - > Pipe0 / DTMODE = 0, to extract one flow
    - > Pipe0 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the flow to dump for instance
- Sensors in 3D, with left and right image inside a unique global image:
  - If the left and right images are differentiated with a left and right part of a single image, that double image can be extracted and duplicated to both the pixel pipes, and there, cropped differently at input of Pipe1 and Pipe2 respectively
- Sensors with multiple DT:
  - The sensor is supported by sending all (or most) of the flows to the dump pipe, and dumped consecutively, inserting ahead of each dumped packet an header containing the VC, DT, and packet length. See *Section 42.7.8* for more details on the header format. One or two pixel flows can still be extracted and processed by Pipe1 and Pipe2, and potentially suppressed from the extraction of the data dump, thanks to DCMIPP_PxFSCR.DTMODE.

42.9.3 Force data type format from CSI-2 data flow (pixel pipes only)

Application may require to use a user data type in the CSI-2 data flow without specifying the format used for the pixel transfer from a camera bridge to the DCMIPP, thanks to CSI-2 interface physical link.

However, the pixel pipes need to know the data format to process them in a proper way along the pipeline. It is possible to force a data type by software thanks to bits FDTF[5:0] and FDTFEN[5:0]. Only the pixel pipes (Pipe1 and Pipe2) can benefit of this capability. Indeed, Pipe0 manages the data and not the pixel itself.

If FDTFEN bit is set in the DCMIPP_PxFSCR register, the data type set into DTIDA/DTIDB (coupled with DTMODE) is used to detect the data type to be overwritten by FDTF[5:0] value. The value used to force the data type must follow CSI-2 data type format specification to identify the CSI-2 normalized data format.

Note: The forced data type value is unique. This means that, if two data types have been selected to be propagated into Pipe1 (DTMODE = 1), both DTIDA and DTIDB detected in input of the pipe are considered by Pipe1 with the forced value FDTF[5:0] if bit FDTFEN is set.

If the virtual channel and data type values are identical between Pipe1 and Pipe2, the force value FDTF[5:0] needs to be aligned within these two pipes if the respective FDTFEN bits are set, otherwise the behavior is not guaranteed.

If PIPEDIFF = 1, any data type format linked to a raw format can be forced on Pipe2 since it is not sharing the demoisaicing block of Pipe1.

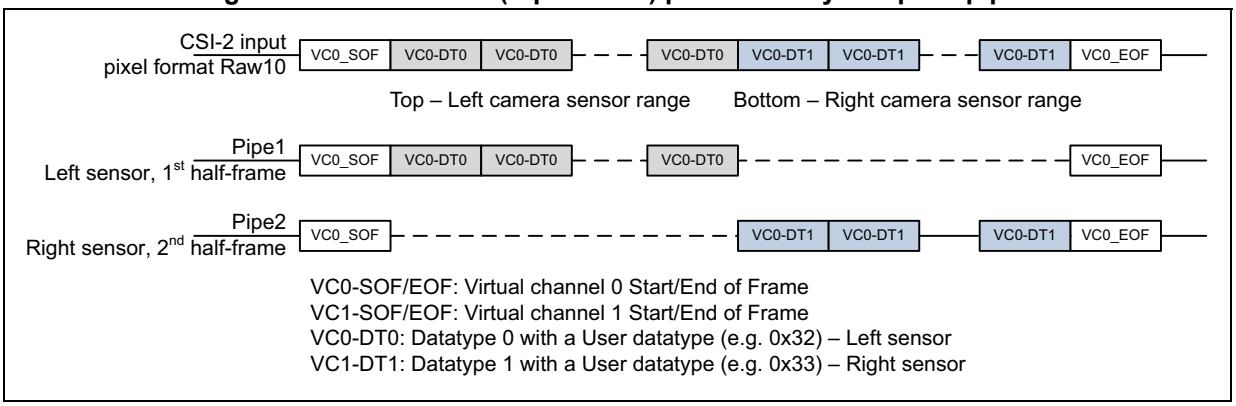
Typical application example: 3D sensors camera (top-bottom)

Some CSI-2 bridges concatenate data from left and right camera in a top-bottom approach. It means that within a frame, first half of the lines are the ones generated by the left camera, whereas the half last lines of a frame are attached to the right camera sensor. The bridge may typically make a distinction between lines coming from left camera and right camera by identifying the data type with a user data type value linked to a camera.

The user data type does not identify a pixel format, but rather a data linked to a camera. To help the pipeline process in a correct way the pixel, the software must inform the pipeline about the pixel format in input attached to the user Datatypes.

Let us imagine that the application has to separate pixels from the left camera from those of the right camera. Two pixel pipes can be configured for that, with, for instance, Pipe1 for left camera and Pipe2 dedicated to extract pixels from the right camera, as shown in *Figure 360*.

Figure 360. 3D sensors (top-bottom) processed by two pixel pipes



DCMIPP_P1FSCR configuration

- PIPEDIFF = 0, Pipe1 shares the demosaicing block with Pipe2 to transform, for instance, Raw10 in RGB888 for the output pixel format.
- VC = Virtual channel number
- DTMODE = 1, both Datatypes (0x32 and 0x33) selected to be propagated into Pipe1 through the demoisaicing, shared with Pipe2.
- DTIDA = 0x32, to let pixel data from the left camera sensor (identified by the user data type 0x32) to be processed by the demosaicing block, and further Pipe1 post processing stages (crop, downsize).
- DTIDB = 0x33, to let pixel data from the right camera sensor (identified by the user data type 0x33) to be processed by the demosaicing block, and further Pipe2 post processing stages (crop, downsize).
- FDTF[5:0] = 0x2B, to inform about the Raw10 data type normalized value in CSI-2 specification as input pixel format of the pipes.
- FDTFEN = 1, to force the user Datatypes DTIDA and DTIDB to be decoded as FDTF[5:0] input pixel format value.
- PIPEN = 1, to activate Pipe1.

Pipe1 crop registers (DCMIPP_P1CRSTR and DCMIPP_P1CRSZR) must be configured to crop only the first half lines of the frame corresponding to the pixels coming from the left camera sensor.

DCMIPP_P2FSCR configuration:

- VC / DTIDA / DTIDB / FDTF[5:0] / FDTFEN = don’t care since it is the same as the one defined in DCMIPP_P1FSCR (as PIPEDIFF = 0).
- PIPEN = 1 to activate Pipe2.

Pipe2 crop registers (DCMIPP_P2CRSTR and DCMIPP_P2CRSZR registers) must be configured to crop only the second half lines of the frame corresponding to the pixels coming from the right camera sensor.

42.10 Pixel format description

This section describes the pixel formats used in the parallel interface and in the pixel pipes and dump pipe. The CSI-2 pixel format is described in the CSI-2 v1.3 Standard.

The support for the pixel formats, per interface (parallel, CSI-2, pixel output, dump output) are summarized in *Table 403*, *Table 379*.

Note: In the following tables the components are represented with their symbolic color: red and Cr (= V) in red, blue and Cb (= U) in blue, green in green, and Y, monochrome, and all raw Bayer components in gray.

42.10.1 Parallel interface formats

This paragraph describes the input pixel format as supported by the parallel interface of the DCMIPP and the possible swap combinations.

Note: The parallel interface does not input specifically RGB444/RGB555 (and RGB666). However, a sensor with these output can connect them onto the DCMIPP, by selecting RGB565 (and RGB888), and by either connecting the missing bits with the MSB of the sensor output or by strapping them.

Table 403. Parallel interface input pixel formats

Index	IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
13	Byte	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
5	Raw Bayer 8	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
6	Raw Bayer 10	Cycle 1/1	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
7	Raw Bayer 12	Cycle 1/1	-	-	-	-	11	10	9	8	7	6	5	4	3	2	1	0
8	Raw Bayer 14	Cycle 1/1	-	-	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	Monochrome 8	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
10	Monochrome 10	Cycle 1/1	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
11	Monochrome 12	Cycle 1/1	-	-	-	-	11	10	9	8	7	6	5	4	3	2	1	0
12	Monochrome 14	Cycle 1/1	-	-	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2-1	RGB565 - 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
		Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
2-2	RGB565 - 16 bits	Cycle 1/1	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
2-2	PCB: Sensor444 to DCMIPP-RGB565	Cycle 1/1	R3	R2	R1	R0	R3	G3	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3
2-2	PCB: Sensor555 to DCMIPP-RGB565	Cycle 1/1	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	G4	B4	B3	B2	B1	B0
4-1	RGB888 - 12 bits	Cycle 1/2	-	-	-	-	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4
		Cycle 2/2	-	-	-	-	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
	YUV444 -12 bits	Cycle 1/2	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4
		Cycle 2/2	-	-	-	-	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0

Table 403. Parallel interface input pixel formats (continued)

Index	IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4-2	RGB888 - 8 bits	Cycle 1/3	-	-	-	-	-	-	-	-	R7	R6	R5	R4	R3	R2	R1	R0
		Cycle 2/3	-	-	-	-	-	-	-	-	G7	G6	G5	G4	G3	G2	G1	G0
		Cycle 3/3	-	-	-	-	-	-	-	-	B7	B6	B5	B4	B3	B2	B1	B0
	YUV444 - 8 bits	Cycle 1/3	-	-	-	-	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0
		Cycle 2/3	-	-	-	-	-	-	-	-	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Cycle 3/3	-	-	-	-	-	-	-	-	U7	U6	U5	U4	U3	U2	U1	U0
3	PCB: Sensor 666 to RGB888	Cycle 1/3	-	-	-	-	-	-	-	-	R5	R4	R3	R2	R1	R0	R5	R4
		Cycle 2/3	-	-	-	-	-	-	-	-	G5	G4	G3	G2	G1	G0	G5	G4
		Cycle 3/3	-	-	-	-	-	-	-	-	B5	B4	B3	B2	B1	B0	B5	B4
1-1	YUV422 - 8 bits (YUY'V)	Cycle 1/4	-	-	-	-	-	-	-	-	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Cycle 2/4	-	-	-	-	-	-	-	-	U7	U6	U5	U4	U3	U2	U1	U0
		Cycle 3/4	-	-	-	-	-	-	-	-	Y7'	Y6'	Y5'	Y4'	Y3'	Y2'	Y1'	Y0'
		Cycle 4/4	-	-	-	-	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0
1-2	YUV422 - 16 bits (YUY'V)	Cycle 1/2	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0
		Cycle 2/2	Y7'	Y6'	Y5'	Y4'	Y3'	Y2'	Y1'	Y0'	V7	V6	V5	V4	V3	V2	V1	V0

Table 404. Correspondence between index and DCMIPP_PRCR register values

Index	FORMAT[7:0]	EDM[2:0]
1-1	0x1E	0x00
1-2	0x1E	0x04
2	0x22	0x04
3	0x23	0x00
4-1	0x24	0x02
4-2	0x24	0x00
5	0x2A	0x00
6	0x2B	0x01
7	0x2C	0x02
8	0x2D	0x03
9	0x4A	0x00
10	0x4B	0x01
11	0x4C	0x02
12	0x4D	0x03
13	Other values	0x00

To adapt to non-standard sensors, the parallel interface is flexible and allows to swap bits, cycles, components, namely:

- Bits: LSB-vs.-MSB bits are swapped, across the whole 16-bit IO pins.
- Cycles: cycle 1/2 and cycle 2/2 are swapped. No impact for 1 or 3 cycles.
- Components: R-vs-B components are swapped (or V-vs-U if YUV).

Table 405 shows these permutations, based on the RGB565 format. It lists the sampled bits of cycles 1 and 2, on pins 0 to 7, and for each of them returns the assignation of the RGB output components.

Table 405. Parallel interface input pixel formats

IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NATIVE RGB565 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
	Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
SWAP LSB-MSB RGB565 - 8 bits	Cycle 1/2	G3	G4	G5	R0	R1	R2	R3	R4	-	-	-	-	-	-	-	-
	Cycle 2/2	B0	B1	B2	B3	B4	G0	G1	G2	-	-	-	-	-	-	-	-
SWAP cycles RGB565 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
	Cycle 2/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
SWAP R-vs-B RGB565 8 bits (not available on dump pipe)	Cycle 1/2	-	-	-	-	-	-	-	-	B4	B3	B2	B1	B0	G5	G4	G3
	Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	R4	R3	R2	R1	R0

42.10.2 Pixel pipe formats

Table 406. Pixel pipe output pixel format(1)

Format	Memory bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RGB888 (packed)	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
		Pixel N + 1								Pixel N + 0																							
		G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0
		Pixel N + 2																Pixel N + 1															
		R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0
		Pixel N + 3																								Pixel N + 2							
1	RGB565	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
		Pixel N + 1																Pixel N + 0															
2	xRGB888 (32 bpp)	x (Filling = 0xFF)								R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
		Pixel N																															
3	RGBx888 (32 bpp)	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	x (filling = 0xFF)							
		Pixel N																															

Table 406. Pixel pipe output pixel format(1) (continued)

Format	Memory bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	Mono or raw Bayer (8 bpp)	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
		Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
5	AYUV888 (32 bpp, AYUV)	A (Filling = 0xFF)								Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0
		Pixel N																															
6	YUV422-1 buffer (YUYV)	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Pixel N + 1																Pixel N + 0															
7	YUV422-2 buffers (-)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 2 and 3																Pixel N + 0 and 1															
8	YUV420-2 buffers (NV21, NV12 if SwapRB)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 2 and 3																Pixel N + 0 and 1															
9	YUV420-3 buffers (YV12)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 6 and 7								Pixel N + 4 and 5								Pixel N + 2 and 3								Pixel N + 0 and 1							
		V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0
		Buffer2: Pixel N + 6 and 7								Pixel N + 4 and 5								Pixel N + 2 and 3								Pixel N + 0 and 1							
10	YUV422-1 buffer (UYVY)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0
		Pixel N + 1																Pixel N + 0															

1. Format identifies the value set in bitfield FORMAT[3:0] of DCMIPP_PxPPCR registers (x = 1 to 2).

42.10.3 Dump pipe formats

The dump pipe, thanks to its capability to dump, supports more formats than the pixel pipe, namely:

- CSI-2 packet headers and data
- CSI-2 raw Bayer pixels on 6/7bits (that can be dumped as-is but are not supported by the pixel pipe and hence cannot be processed as pixels).
- Monochrome and raw Bayer pixels on 10/12/14 bits, padded on 16 bpp.

Table 407 details the support for these additional pixel formats. Other formats are assumed to be pixels with the same width. The other pixel formats used by the dump pipe are similar to the pixel formats used by the pixel pipe.

Table 407. Dump pipe OUTPUT pixel formats(1)

Bits	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
CSI-2 header	Packet data (FrameNumber or ByteCount)	DataType-ID	VC-ID	FrameType
	Header 32 bits			
CSI-2 data	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
	Data 32 bits			
CSI-2 raw Bayer 6 bpp (words 0 to 2)	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0	5 4 3 2 1 0
	PN + 5 / PN + 4	PN + 3	PN + 2	PN + 1 / PN + 0
	3 2 1 0 5 4 3 2 1 0 5 4 3 2	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2
	PN + 11 / PN + 10	PN + 9	PN + 8	PN + 7 / PN + 6
	5 4 3 2 1 0 5 4 3 2 1 0 5 4	3 2 1 0 5 4 3 2 1 0 5 4 3 2	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4
	PN + 16	PN + 15	PN + 14	PN + 13 / PN + 12 / PN +11
CSI-2 raw Bayer 7 bpp (words 0 to 7)	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4
	PN + 4	PN + 3	PN + 2	PN + 1 / PN + 0
	31 6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4
	PN + 9 / PN + 8	PN + 7	PN + 6	PN + 5 / PN + 4
	4 3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1
	PN + 13	PN + 12	PN + 11	PN + 10 / PN + 9
	1 0 6 5 4 3 2 1 0 6 5 4 3 2	1 0 6 5 4 3 2 1 0 6 5 4 3 2	1 0 6 5 4 3 2 1 0 6 5 4 3 2	1 0 6 5
	PN + 18 / PN + 17	PN + 16	PN + 15	PN + 14 / PN +13
	5 4 3 2 1 0 6 5 4 3 2 1 0 6	5 4 3 2 1 0 6 5 4 3 2 1 0 6	5 4 3 2 1 0 6 5 4 3 2 1 0 6	5 4 3 2
	PN + 22	PN + 21	PN + 20	PN + 19 / PN + 18
	2 1 0 6 5 4 3 2 1 0 6 5 4 3	2 1 0 6 5 4 3 2 1 0 6 5 4 3	2 1 0 6 5 4 3 2 1 0 6 5 4 3	2 1 0 0
	PN + 27	PN + 26	PN + 25	PN + 24 / PN + 23
	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3
	PN + 31	PN + 30	PN + 29	PN + 28 / PN + 27
Byte/ mono/raw 8 bpp	7 6 5 4 3 2 1 0 7 6 5 4 3 2	1 0 7 6 5 4 3 2 1 0 7 6 5 4	3 2 1 0 7 6 5 4 3 2 1 0 7 6	5 4 3 2 1 0
	PN + 3	PN + 2	PN + 1	PN + 0

Table 407. Dump pipe OUTPUT pixel formats(1) (continued)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAD = 0 Mono/raw 10 bpp	0						9	8	7	6	5	4	3	2	1	0	0						9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
PAD = 0 Mono/raw 12 bpp	0				11	10	9	8	7	6	5	4	3	2	1	0	0				11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
PAD = 0 Mono/raw 14 bpp	0	13	12	11	10	9	8	7	6	5	4	3	2	1	0		0	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	PN + 1																PN + 0															
PAD = 1 Mono/raw 10 bpp	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0
	PN + 1																PN + 0															
PAD = 1 Mono/raw 12 bpp	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0
	PN + 1																PN + 0															
PAD = 1 Mono/raw 14 bpp	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0
	PN + 1																PN + 0															
RGB 16 bpp	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
RGB 24 bpp (words 0 to 2)	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1								PN + 0																							
	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
	PN + 2																PN + 1															
	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16
	PN + 3																						PN + 2									
	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0
YUV422-1 (1 word)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
	Y'7	Y'6	Y'5	Y'4	Y'3	Y'2	Y'1	Y'0	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0

1. PN indicates Pixel N.

42.10.4 AXI IP-Plug

The AXI IP-Plug dumps the data of the pipes (via the AXI) to the memory. It is unique, shared by all the pipes.

Features

- Pipe ID: AXI IP-Plug differentiates, via its ID, the source of the pipe. It allows to differentiate the QoS depending upon the clients.
- FIFOs: sized to handle up to 5 µs latencies (1 µs average round-trip + 4 µs memory slowdown), with, by default, at reset:
  a) Pipe0: 5 Gbps, thus up to 4 Kbytes (if ARGB)
  b) Pipe1-Y/RGB: 200 Mpixel/s peak x 8bpp / 32bpp, thus up to 4 Kbytes
  c) Pipe1-U: 200 Mpixel/s peak x 8 bpp, thus up to 1 Kbytes
  d) Pipe1-V: 200 Mpixel/s peak x 4 bpp, thus up to 0.5 Kbytes
  e) Pipe2: 200 Mpixel/s peak x 32 bpp, thus up to 4 Kbytes
  f) Stats: 320 bins of 16 bpp, thus 640 bytes
- Burst length: 64 and 128 bytes for all the pipes.
- Outstanding capability: 16 outstandings, to cope with 2 GB/s maximum bandwidth on 128 bytes bursts.
- Bandwidth limiter (via an outstanding limitation per pipe): avoids a last-line dump that would uselessly overload the memory bandwidth.
- Unique AXI-QOS: unique QOS for the DCMIPP (without differentiation between pipes that must all work correctly with their real-time constraint).
- The total FIFO size is 12160 bytes.

Common software configuration

The AXI IP-Plug has a common configuration in the DCMIPP_IPGR1/2/3/8 registers.

The common registers are writable only when the IP-Plug is in Idle mode (see PSTART below to lock it, or below for an example).

- MEMORYPAGE: for efficient handling of DRAM: bursts do not cross this module.
- PSTART: request to freeze the plug, to reconfigure it safely.
- IDLE: status to mention the plug is frozen, to reconfigure it safely.
- IPPID[7:0], ARCHIID[4:0], REVID[4:0], DID[5:0]: version IDs of the IP-Plug.

Per-client software configuration

The AXI IP-Plug has a per-client configuration in the registers DCMIPP_IPCxR1/2/3.

The client registers are writable only when the IP-Plug is in Idle mode (see PSTART in common registers above to lock it, or below for an example).

The clients (1..n) are numbered following the sequence Pipe0..m. Inside each pipe, there are several potential clients when the planar feature is needed.

For instance, the DCMIPP with its three pipes (Pipe0, Pipe1, Pipe2) and with Pipe1 allowing a full planar capability (three buffers: Y, U, V), has the IP-Plug clients, numbered as follows:

- Client1 = Pipe0
- Client2 = Pipe1Y/RGB
- Client3 = Pipe1U
- Client4 = Pipe1V
- Client5 = Pipe2
- Client6 = Statistics (dump of the histograms)

The configurable per-client features are the following:

- OTR: outstanding transactions per client (Outstanding = OTR + 1)
- TRAFFIC[2:0]: burst size: best select 128 bytes, thus TRAFFIC = 4
- WLRU[3:0]: proportion of total bandwidth: WLRU = BWi / sum (all BW)
- SVC: system virtual channel, keep SVC = 0
- DPREGSTART: local base-address of start of FIFO for this client
- DPREGEND: local base-address of end of FIFO (included word) for this client

Configuration computations

IP-Plug common configuration:

- PSTART: set to 1, to put IP-Plug idle and configure its registers G1/3, R1/3
- IDLE: read and wait till 1, to get it idle
- MEMORYPAGE: 4, to optimize for memory pages of 256 bytes

IP-Plug per-client configuration:

- OTR (client): outstanding per client, among a total of 16 outstandings.
  - The provided value (OTR) is the outstanding amount minus 1.
- DPREG size (client): RAM shared among client, among a total of 1520 x 64-bit words (80 words for the histogram)
  - DPREGSTART = 0 (or after another client location).
  - DPREGEND = 127 for ((127 - 0) + 1) x 8 B = 1 KB, as example if 1 KB is assigned to this client. The value to program is 127, as the word is included. DPREGEND is defined to include the last given word of memory, so that a FIFO of 128 words (starting at 0) has DPREGEND = 127.
- WLRU (client): Weighted Least Recently Used arbitration ratio, represents the bandwidth ratio for each client. The maximum ratio WLRU value is 16, meaning that the most demanding client will have 16 times the bandwidth of the least demanding client. The value to configure in DCMIPP_IPCxR2 is the WLRU ratio minus 1.

Each client is provided a proportion of the total FIFO size and of the total outstanding available transactions, in function of the local peak bandwidth. Indeed, the DCMIPP implements buffers to smooth the bandwidth, but they are not large enough to buffer full lines. The proportion of the peak bandwidth of each pipe is estimated as:

- PeakBandwidthProportion (= PBP) = OutputWidth (= OW) * PixelFormatSize (= PFS)
- OTR(client) = Total_OTR * PBP(client) / SumAll(PBP) - 1
- DPREG_Size(client) = Total_DPREG_Size * PBP(client) / SumAll(PBP)

As an example, an xRGB 720p display output, a VGA 422 video output (for software), and no statistics, results in:

- PBP(Display) = 1280 * 4 = 5120
- PBP(Video) = 640 * 2 = 1280
- OTR(Display) = 16 * 5120 / (5120 + 1280) - 1 = 12
- OTR(Video) = 16 * 1280 / (5120 + 1280) - 1 = 3
- DPREG_Size(Display) = 1440 * 5120 / (5120 + 1280) = 1152
- DPREG_Size(Video) = 1440 * 1280 / (5120 + 1280) = 288

Video is assigned to Pipe1 (= Client2) and display to Pipe2 (Client5):

- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfiguration)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Video as Pipe1, statically mapped onto IP-Plug Client2:
  - DCMIPP_IPC2R1.OTR = 3
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default)
  - DCMIPP_IPC2R3.DPREGSTART = 1152
  - DCMIPP_IPC2R3.DPREGEND = 1439
- Display as Pipe2, statically mapped onto IP-Plug Client5:
  - DCMIPP_IPC5R1.OTR = 12
  - DCMIPP_IPC5R1.TRAFFIC = 4 (default)
  - DCMIPP_IPC5R3.DPREGSTART = 0
  - DCMIPP_IPC5R3.DPREGEND = 1151
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

Configuration example 1: Single RGB dump

It is the simplest but most usual configuration, with the DCMIPP dumping to a single xRGB buffer. It has thus a single pipe active, the dump is coplanar (no side U or V planes), and to emphasize, it does not dump separate statistics (with Pipe0), nor does use the second pixel pipe (Pipe2).

All the AXI capabilities are focused on the single active pipe, which gets all the FIFO space and outstanding capability of the DCMIPP. With this setting, a camera dump of 200 Mpixel/s peak on 32 bpp (800 MB/s bandwidth) benefits of:

- 11 Kbytes FIFO, allowing a memory peak slowdown of 13 µs (a very high value).
- 16 outstandings of 128 bytes, allowing an average memory latency of 2.5 µs (a high value).
- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfig)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Single dump on Pipe1 (main pipe), statically mapped onto IP-Plug Client2:
  - DCMIPP_IPC2R1.OTR = 15 (maximum assigned here)
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC2R2.WLRU = 0 (not used as single client)
  - DCMIPP_IPC2R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC2R3.DPREGSTART = 0 (single FIFO, start at lowest)
  - DCMIPP_IPC2R3.DPREGEND = 1439 (single FIFO, end at highest)
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

Configuration example 2: All pipes active

This is the most flexible configuration of the DCMIPP: the six clients of the IP-Plug are configured to dump the maximum bandwidth they can receive).

Note that, as the bus capabilities are shared among the clients, that configuration is less resistant to memory latencies than the above configuration with only a single RGB dump.

The capabilities of the DCMIPP are shared among the clients with the following assumptions:

- Client1 for Pipe0: dump of raw Bayer pixels, padded to 16 bpp, BW = 2 / 13
- Client2 for Pipe1-Y: dump of ARGB pixels, on 32 bpp, BW = 4 / 13
- Client3 for Pipe1-UV: dump of UV components, on 8 bpp, BW = 1 / 13
- Client4 for Pipe1-V: dump of V component, on 4 bpp, BW = 1 / 13
- Client5 for Pipe2: dump of ARGB pixels, on 32 bpp, BW = 4 / 13
- Client6 for Stats: dump of 320 bins, on 16 bpp, BW = 1 / 13

The total bandwidth proportion, when all pipes are used, is 13. As some configurations (like the outstanding capability) are divided in other granularities, Pipe1-Y/RGB has some advantages, sometimes is used alone (as in the above simpler use case).

With these settings, a camera dump of 200 Mpixel/s peak on 32 bpp (800 MB/s bandwidth) benefits, for an average pipe like Pipe2, of:

- 4 Kbytes FIFO, allowing a memory peak slowdown of ~5 µs
- Four outstandings of 128 bytes, with an average memory latency of 0.65 µs, (acceptable but not so large): the DCMIPP must get priority to access the memory.

It results in the following register configuration:

- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfiguration)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Client1 for Pipe0: (raw Bayer, 16 bpp) with 2 / 13 of the BW:
  - DCMIPP_IPC1R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC1R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC1R2.WLRU = 1 (2 parts of BW, among 13)
  - DCMIPP_IPC1R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC1R3.DPREGSTART = 0 (224 parts of FIFO, among 1440)
  - DCMIPP_IPC1R3.DPREGEND = 223 (224 parts of FIFO, among 1440)
- Client2 for Pipe1-Y/RGB: (ARGB, 32 bpp) with 4 / 13 of the BW:
  - DCMIPP_IPC2R1.OTR = 5 (6 outstandings, among 16)
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC2R2.WLRU = 3 (4 parts of BW, among 13)
  - DCMIPP_IPC2R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC2R3.DPREGSTART = 224 (496 parts of FIFO, among 1440)
  - DCMIPP_IPC2R3.DPREGEND = 719 (496 parts of FIFO, among 1440)
- Client3 for Pipe1-UV: (UV, 8 bpp) with 2 / 13 of the BW:
  - DCMIPP_IPC3R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC3R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC3R2.WLRU = 0 (1 part of BW, among 13)
  - DCMIPP_IPC3R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC3R3.DPREGSTART = 720 (144 parts of FIFO, among 1440)
  - DCMIPP_IPC3R3.DPREGEND = 863 (144 parts of FIFO, among 1440)
- Client4 for Pipe1-V: (V, 4 bpp) with 1 / 13 of the BW:
  - DCMIPP_IPC4R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC4R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC4R2.WLRU = 0 (1 part of BW, among 13)
  - DCMIPP_IPC4R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC4R3.DPREGSTART = 864 (80 parts of FIFO, among 1440)
  - DCMIPP_IPC4R3.DPREGEND = 943 (80 parts of FIFO, among 1440)
- Client5 for Pipe2: (ARGB, 32 bpp) with 4 / 13 of the BW:
  - DCMIPP_IPC5R1.OTR = 3 (4 outstandings, among 16)
  - DCMIPP_IPC5R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC5R2.WLRU = 3 (4 parts of BW, among 13)

  - DCMIPP_IPC5R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC5R3.DPREGSTART = 944 (496 parts of FIFO, among 1440)
  - DCMIPP_IPC5R3.DPREGEND = 1439 (496 parts of FIFO, among 1440)
- Client6 for statistic histograms (640 bytes):
  - DCMIPP_IPC6R1.OTR = 1 (2 outstandings among 16)
  - DCMIPP_IPC6R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC6R2.WLRU = 0 (1 part of BW, among 13)
  - DCMIPP_IPC6R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC6R3.DPREGSTART = 1440 (80 parts of FIFO)
  - DCMIPP_IPC6R3.DPREGEND = 1519 (80 parts of FIFO)
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

42.11 Shadow registers

A dump and/or pixel pipe must have the capability to be reconfigured without stopping it. Some registers can be written in the middle of a frame without impacting the actual frame acquisition. The values are stored into some shadow registers before being written in their corresponding physical registers, based on a trigger event like described in *Table 408*.

Each physical register in the register map has its address increased by 0x200 respect to its own shadow register address.

Table 408. Shadow and physical registers

Shadow register	Physical register	Trigger event
DCMIPP_P0FSCR(1)	DCMIPP_P0CFSCR	Vsync boundaries
DCMIPP_P0FCTCR	DCMIPP_P0CFCTCR	Vsync boundaries
DCMIPP_P0SCSTR	DCMIPP_P0CSCSTR	Vsync boundaries
DCMIPP_P0SCSZR	DCMIPP_P0CSCSZR	Vsync boundaries
DCMIPP_P0PPCR	DCMIPP_P0CPPCR	Vsync boundaries
DCMIPP_P0PPM0AR1	DCMIPP_P0CPPxM0AR1	Vsync boundaries
DCMIPP_P1FSCR(1)	DCMIPP_P1CFSCR	Vsync boundaries
DCMIPP_P1FCTCR	DCMIPP_P1CFCTCR	Vsync boundaries
DCMIPP_P1BPRCR	DCMIPP_P1CBPRCR	Vsync boundaries
DCMIPP_P1BLCCR	DCMIPP_P1CBLCCR	Vsync boundaries
DCMIPP_P1EXCR1	DCMIPP_P1CEXCR1	Vsync boundaries
DCMIPP_P1EXCR2	DCMIPP_P1CEXCR2	Vsync boundaries
DCMIPP_P1STxCR (x = 1 to 3)	DCMIPP_P1CSTxCR (x = 1 to 3)	Vsync boundaries
DCMIPP_P1STSTR	DCMIPP_P1CSTSTR	Vsync boundaries
DCMIPP_P1STSZR	DCMIPP_P1CSTSZR	Vsync boundaries
DCMIPP_P1HSCR	DCMIPP_P1CHSCR	Vsync boundaries

Table 408. Shadow and physical registers (continued)

Shadow register	Physical register	Trigger event
DCMIPP_P1HSSTR	DCMIPP_P1CHSSTR	Vsync boundaries
DCMIPP_P1HSSZR	DCMIPP_P1CHSSZR	Vsync boundaries
DCMIPP_P1CHSMAR1	DCMIPP_P1CCHSMAR1	Vsync boundaries
DCMIPP_P1CTCRx (x = 1 to 3)	DCMIPP_P1CCTCRx (x = 1 to 3)	Vsync boundaries
DCMIPP_P1CC1CR	DCMIPP_P1CCC1CR	Vsync boundaries
DCMIPP_P1CC1RRx (x = 1 to 2)	DCMIPP_P1CCC1RRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CC1GRx (x = 1 to 2)	DCMIPP_P1CCC1GRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CC1BRx (x = 1 to 2)	DCMIPP_P1CCC1BRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CRSTR	DCMIPP_P1CCRSTR	Vsync boundaries
DCMIPP_P1CRSZR	DCMIPP_P1CCRSZR	Vsync boundaries
DCMIPP_P1DCCR	DCMIPP_P1CDCCR	Vsync boundaries
DCMIPP_P1DSCR	DCMIPP_P1CDSCR	Vsync boundaries
DCMIPP_P1DSRTIOR	DCMIPP_P1CDSRTIOR	Vsync boundaries
DCMIPP_P1DSSZR	DCMIPP_P1CDSSZR	Vsync boundaries
DCMIPP_P1PPCR	DCMIPP_P1CPPCR	Vsync boundaries
DCMIPP_P1PPM0AR1	DCMIPP_P1CPPM0AR1	Vsync boundaries
DCMIPP_P1PPM0PR	DCMIPP_P1CPPM0PR	Vsync boundaries
DCMIPP_P1PPM1AR1	DCMIPP_P1CPPM1AR1	Vsync boundaries
DCMIPP_P1PPM1PR	DCMIPP_P1CPPM1PR	Vsync boundaries
DCMIPP_P1PPM2AR1	DCMIPP_P1CPPM2AR1	Vsync boundaries
DCMIPP_P2FSCR(1)	DCMIPP_P2CFSCR	Vsync boundaries
DCMIPP_P2FCTCR	DCMIPP_P2CFCTCR	Vsync boundaries
DCMIPP_P2CRSTR	DCMIPP_P2CCRSTR	Vsync boundaries
DCMIPP_P2CRSZR	DCMIPP_P2CCRSZR	Vsync boundaries
DCMIPP_P2DSCR	DCMIPP_P2CDSCR	Vsync boundaries
DCMIPP_P2DCCR	DCMIPP_P2CDCCR	Vsync boundaries
DCMIPP_P2DSRTIOR	DCMIPP_P2CDSRTIOR	Vsync boundaries
DCMIPP_P2DSSZR	DCMIPP_P2CDSSZR	Vsync boundaries
DCMIPP_P2PPCR	DCMIPP_P2CPPCR	Vsync boundaries
DCMIPP_P2PPM0AR1	DCMIPP_P2CPPM0AR1	Vsync boundaries
DCMIPP_P2PPM0PR	DCMIPP_P2CPPM0PR	Vsync boundaries

1. Bit PIPEN of this register is not shadowed.

It is mandatory to refresh the shadow registers within a frame to prepare the context to be ready for the next start of frame, to avoid any unpredictable behavior during the acquisition.

The software must then react to the following interrupt sources to launch the shadow registers accesses:

- At VSYNC interrupt (FrameStart at start of pipe) an interrupt is generated, and the software can update the shadow registers loaded into the physical registers at the end of the current frame (so active for the next frame). The software has an entire frame period to complete the update.
- At LINE interrupt line selection can be used also to trigger the software to update the physical registers with the new frame context to consider. The application code must ensure that the update takes place before the end of frame event (current frame). If not, the context can be partially considered from the next frame, dealing with inconsistent settings.

In any case, the software must ensure that the shadow registers are updated before the end of the current frame, to be active from the next frame. Updates started and not completed before the end of current frame result in inconsistent frame acquisition for the next frame (mixing old and new configurations).

It is recommended to change non-shadowed registers when the corresponding pipe is disabled and in an idle state (refer to *Section 42.4.9: Pipe deactivation*).

42.12 DCMIPP low power modes

This section describes the behavior of the DCMIPP versus the modes listed in *Table 409*.

Table 409. DCMIPP low power modes

Mode	Description
Stop	Peripheral content is kept. It is recommended to follow the pipe(s) disabling procedure before entering it.
Standby	Powered-down. The peripherals must be reinitialized after exiting this mode.
Sleep	No effect. DCMIPP is still working in this mode, peripheral interrupts cause the device to exit it.

42.13 DCMIPP interrupts

This section describes the DCMIPP interrupts, and more globally, the features that are related to real-time, namely:

- Free-running: DCMIPP can run without CPU involvement
- Shadow registers: relieve timing constrains from the software
- Interrupts: source of interrupt
- Multiple CPU: capability for independent CPU or driver without Mutex
- Event pins: to drive external IPs, like a DMA

42.13.1 Free-running DCMIPP

After a camera flow has been configured and established, it is able to run permanently, without any software involvement. The following features are available:

- Permanent pipe activation, for Pipe0, 1, 2: when set, the respective pipe dumps frame data, for an unlimited amount of frames.
  - DCMIPP_PxFCTCR.CPTMODE = 0
  - DCMIPP_PxSR.CPTACT = 1
  - DCMIPP_PxFCTCR.FRATE = xx (depending on desired frame subsampling rate)
- No double buffered frames: the DCMIPP defines only one BaseAddress pointer for the output buffers:
  - Software can react to the VSYNC interrupt, and can decide, before the next VSYNC event, to modify the frame BaseAddress pointer.
  - If the software does not react, the capture of a frame overwrites the previous one.

42.13.2 Interrupts

The DCMIPP handles five interrupts pins, the fifth being the OR of the first four.

A register set is associated to each interrupt to provide the following functionality:

- Unmasked events: reports an event that occurred and has not yet been cleared
- Masks: to mask unwanted events
- Masked events: not readable, any active unmasked bit triggers the interrupt pin
- Clear register: write to clear any active event

The interrupt pins are active when high: they go high when triggered by an unmasked event, and remain high until all the masked events of that interrupt are cleared.

The interrupts lines and registers are:

1. Parallel interface, with the next interrupt events handled:
   - ERR: bad embedded synchronization detected.
     > Unexpected behavior in the parallel interface, usually not triggered if no issues.
     > It is due to an unexpected sensor behavior, or to a transmission error.
2. Pipe0 (dump), with the next interrupt events handled:
   - OVR: data overflow: the memory was too slow vs. received pixels.
     > Unexpected behavior in Pipe0, usually not triggered if no issues.
     > Same recommendation as for above OVR: skip the impacted frame.
   - LIMIT: received volume is larger than the maximum allowed dump volume.
     > Unexpected behavior in Pipe0, usually not triggered if no issues.
     > The transmitted flow is too long. More space must be allocated in memory to store the transmission of a next frame.
   - VSYNC: main interrupt, where most software can sit.
     > Trigger: at VSync (mid blanking), permanent (even if no dump active).
     > Typically to reconfigure slowly the pipes (at least shadow registers) for next frame
     > Typically to trigger usage of the previously captured frame.
   - FRAME: secondary interrupt, as backup for fast-software
     > Trigger: after last data dump of this pipe, inactive if no capture has occurred.
     > Typically to quickly reconfigure non-shadow registers (during only vertical blanking)
     > Typically to trigger usage of the previously captured frame.
   - LINE: interruption for stripe-based operators trigger: after every 1/2/4/8/16/32/64/128 dumped lines and last line. It is measured and extracted at end of pipe, close to output to memory.
     > Typically to trigger fast/reactive software or hardware stripe-based operators.

Note: The LINE flag (LINEF bit) is set at the end of frame, to trigger out software event when the frame height is not a multiple of the selected number of lines for which the LINEF event is triggered. The last part of the frame can, in such case, have a lower number of lines than the selection. The event/interrupt is generated even if the number of lines is incomplete. The software must consider that this last event has a lower than expected number of lines when reaching the end of frame (FRAME event).

3. Pipe1 (main pipe)
   - Same events handled as Pipe0: OVR, VSYNC, FRAME, LINE, but no LIMIT as only pixels are captured after a potential crop.
   - STFRAME: Histogram statistics frame interrupt
     > Trigger: Histogram statistics captured and successfully written to system memory.
   - STOVR: Histogram statistics overrun interrupt
     > Trigger: The memory was too slow to receive and store the entirety of the histogram data. Statistics capture is aborted.
   - STBC: Histogram statistics bad configuration interrupt

> Trigger: An incoherent histogram configuration was detected. Statistics capture is aborted.

4. Pipe2 (ancillary pipe)
   – Same events handled as Pipe1: OVR, VSYNC, FRAME, LINE.
5. Common interrupt: groups the events of the above interrupts in a single register set and drives its own interrupt line.
   – The fifth interrupt is designed for systems where a single driver handles the whole DCMIPP: a single access to this interrupt status allows to retrieve the status of the whole DCMIPP.
   – The first four interrupts are designed for systems where a driver handles a pipe and (potentially) another one drives the parallel interface. In that case, each driver handles its own individual interrupt register set, without need for semaphores to arbitrate conflicting accesses.

   There is also a common interrupt that can be generated if enabled, when a transfer error is detected during an AXI transfer from the IP-Plug to memories. Error flag ATXERR in the DCMIPP_CMSR2 register is set and an interrupt can be generated to inform the software about this error status. There is no specific hardware action linked to this error, it is up to software to handle the transfer error situation.

   On multiple-pipe architecture, the error informs that at least one AXI client has a transfer error, but does not identify from which client.

The DCMIPP interrupts are summarized in *Table 410*. An event that generates an interrupt does it at two locations:

– once in the common interrupt register set (when driven by a single CPU)
– once in the local interrupt register set (when driven by multiple CPUs).

Table 410 provides the local (register and bit) and global (bit only, as all bits are in the same DCMIPP_CMSR2 register) locations.

Table 410. DCMIPP interrupts

Interrupt event	Local register	Local bit	Common bit in DCMIPP _CMSR2	Event flag/interrupt clearing method	Interrupt enable control bit local/common register
Synchronization error in parallel interface	DCMIPP _PRSR	ERRF	PRERRF	Write CERRF = 1 / CPRERRF = 1	ERRIE/PRERRFIE
AXI transfer error	-	-	ATXERR	Write CATXERRF = 1	ATXERRIE
Overflow in Pipe0	DCMIPP _P0SR	OVRF	P0OVRF	Write COVRF = 1 / CP0OVRF = 1	OVRIE/P0OVRIE
Limit violation in Pipe0		LIMITF	P0LIMITF	Write CLIMITF = 1 / CP0LIMITF = 1	LIMITIE/P0LIMITIE
Frame start (VSYNC) in Pipe0		VSYNC	P0VSYNC	Write CVSYNCF = 1 / CP0VCSYNCF = 1	VSYNCIE/P0VSYNCIE
Frame end (FRAME) in Pipe0		FRAMEF	P0FRAMEF	Write CFRAMEF = 1 / CP0FRAMEF = 1	FRAMEIE/P0FRAMIE
Multi-line (LINE) in Pipe0		LINEF	P0LINEF	Write CLINEF = 1 / CP0LINEF = 1	LINEFIE/P0LINEFIE

Table 410. DCMIPP interrupts (continued)

Interrupt event	Local register	Local bit	Common bit in DCMIPP_CMSR2	Event flag/interrupt clearing method	Interrupt enable control bit local/common register
Overflow in Pipe1	DCMIPP_P1SR	OVRF	P1OVRF	Write COVRF = 1 / CP1OVRF = 1	OVRIE/P1OVRIE
Frame start (VSYNC) in Pipe1		VSYNC	P1VSYNC	Write CLIMITF = 1 / CP1LIMITF = 1	LIMITIE/P1LIMITIE
Frame end (FRAME) in Pipe1		FRAMEF	P1FRAMEF	Write CVSYNCF = 1 / CP1VCSYNCF = 1	VSYNCIE/P1VSYNCIE
Multi-line (LINE) in Pipe1		LINEF	P1LINEF	Write CFRAMEF = 1 / CP1FRAMEF = 1	FRAMEIE/P1FRAMIE
Overflow in Pipe2	DCMIPP_P2SR	OVRF	P2OVRF	Write COVRF = 1 / CP2OVRF = 1	OVRIE/P2OVRIE
Frame start (VSYNC) in Pipe2		VSYNC	P2VSYNC	Write CLIMITF = 1 / CP2LIMITF = 1	LIMITIE/P2LIMITIE
Frame end (FRAME) in Pipe2		FRAMEF	P2FRAMEF	Write CVSYNCF = 1 / CP2VCSYNCF = 1	VSYNCIE/P2VSYNCIE
Multi-line (LINE) in Pipe2		LINEF	P2LINEF	Write CFRAMEF = 1 / CP2FRAMEF = 1	FRAMEIE/P2FRAMIE
Histogram statistics complete	DCMIPP_P1SR	STFRAMEF	STFRAMEF	Write CSTFRAMEF = 1	STFRAMEIE
Histogram statistics overrun		STOVRF	STOVRF	Write CSTOVRF = 1	STOVRIE
Histogram statistics bad configuration		STBCF	STBCF	Write CSTBCF = 1	STBCIE

42.13.3 Event pins

Event pins are exposed to let an ancillary HW IP, like a central DMA, to synchronize with them. The event pins are the unmasked synchronization events (VSYNC, FRAME, LINE, HSYNC) of the pipes.

An internal events generates a pulse (15 APB cycles long) on its pin.

Table 411. Event connection

Pipe	Event name	Internal signal
Pipe0	HSYNC	dcmipp_p0_hsync_evt
	VSYNC	dcmipp_p0_vsync_evt
	Frame	dcmipp_p0_frameend_evt
	Line	dcmipp_p0_lineend_evt

Table 411. Event connection (continued)

Pipe	Event name	Internal signal
Pipe1	HSYNC	dcmipp_p1_hsync_evt
	VSYNC	dcmipp_p1_vsync_evt
	Frame	dcmipp_p1_frameend_evt
	Line	dcmipp_p1_lineend_evt
Pipe2	HSYNC	dcmipp_p2_hsync_evt
	VSYNC	dcmipp_p2_vsync_evt
	Frame	dcmipp_p2_frameend_evt
	Line	dcmipp_p2_lineend_evt

42.14 DCMIPP registers

The registers are split into groups with a same prefix, as shown in *Table 412*.

Table 412. DCMIPP registers organization

Register name	Offset	Function
DCMIPP_IP	0x000	IP-Plug registers
DCMIPP_PR	0x100	Parallel interface
DCMIPP_CM	0x200	Common registers
DCMIPP_P0	0x500	Pipe0 (dump pipe) with shadow registers
	0x700	Pipe0 (dump pipe) with physical registers
DCMIPP_P1	0x800	Pipe1 image processing with shadow registers
	0x900	Pipe1 post-processing with shadow registers
	0xA00	Pipe1 image processing with physical registers
	0xB00	Pipe1 post-processing with physical registers
DCMIPP_P2	0xC00	Pipe2 image processing with shadow registers
	0xD00	Pipe2 post-processing with shadow registers
	0xE00	Pipe2 image processing with physical registers
	0xF00	Pipe2 post-processing with physical registers
DCMIPP_	0xFF0	IP version and configuration (without prefix)

42.14.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1)

Address offset: 0x000

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	QOS_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMORYPAGE[2:0]		
													rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **QOS_MODE:** Quality of service
Set of functions enabling to build and configure an architecture meeting bandwidth and latency requirements.

Bits 23:3 Reserved, must be kept at reset value.

Bits 2:0 **MEMORYPAGE[2:0]:** Memory page size, as power of 2 of 64-byte units:
0x0: 64 bytes
0x1: 128 bytes
0x2: 256 bytes
0x3: 512 bytes
0x4: 1K bytes
0x5: 2K bytes
0x6: 4K bytes
0x7: 8K bytes

42.14.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTART
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PSTART:** Request to lock the IP-Plug, to allow reconfiguration.
0: No lock requested, IP-Plug runs on demand by background HW.
1: Lock requested: IP-Plug freezes shortly (see IDLE bit when lock is active).
PSTART must be reset to 0 after configuration is completed, to restart the IP-Plug.

42.14.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3)

Address offset: 0x008

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLE
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **IDLE:** Status of IP-Plug
0: IP-Plug is running (on demand by background HW)
1: IP-Plug is currently locked and can be reconfigured
IDLE is set after a request by setting PSTART at 1, and reset by resetting PSTART at 0.

42.14.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8)

Address offset: 0x01C

Reset value: 0xAA04 0314

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPPID[7:0]								Res.	Res.	Res.	ARCHIID[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	REVID[4:0]					Res.	Res.	DID[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:24 **IPPID[7:0]:** IP identifier (0xAA)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **ARCHIID[4:0]:** Architecture identifier (0x04)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **REVID[4:0]:** Revision identifier (0x03)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DID[5:0]:** Division identifier (0x14)

42.14.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1)

Address offset: 0x020 + 0x10 * (x - 1), (x = 1 to 6)

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC[2:0]		
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **OTR[3:0]:** Maximum outstanding transactions
0: Disabled. No outstanding transaction limitation (except via FIFO size)
1: Maximum two outstanding transactions ongoing.
...
15: Maximum 16 outstanding transactions ongoing.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TRAFFIC[2:0]:** Burst size as power of 2 of 8-byte units
0x0: 8 bytes
0x1: 16 bytes
0x2: 32 bytes
0x3: 64 bytes
0x4: 128 bytes
Other values: Reserved

42.14.6 DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2)

Address offset: 0x024 + 0x10 * (x - 1), (x = 1 to 6)

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **WLRU[3:0]:** Ratio for WLRU[3:0] arbitration
A client gets a portion of the total bandwidth = Ratio(client) / Sum(all ratios)
0x0: Ratio part = 1
0x1: Ratio part = 2
...
0xF: Ratio part = 16

Bits 15:0 Reserved, must be kept at reset value.

42.14.7 DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3)

Address offset: 0x028 + 0x10 * (x - 1), (x = 1 to 6)

Reset value: 0x017F 0000, 0x02FF 0180, 0x03BF 0300, 0x041F 03C0, 0x059F 0420, 0x05EF 05A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DPREGEND[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DPREGSTART[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **DPREGEND[10:0]:** End word (AXI width = 64 bits) of the FIFO of Clientx.
The addressed word is included in the FIFO, so that next DPREGSTART is DPREGEND + 1.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **DPREGSTART[10:0]:** Start word (AXI width = 64 bits) of the FIFO of Clientx.

42.14.8 DCMIPP parallel interface control register (DCMIPP_PRCR)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SWAP BITS	SWAP CYCLES	Res.	FORMAT[7:0]							
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	EDM[2:0]			Res.	Res.	VSPOL	HSPOL	PCKPOL	ESS	Res.	Res.	Res.	Res.
	rw		rw	rw	rw			rw	rw	rw	rw				

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SWAPBITS**: Swap LSB vs. MSB within each received component
0: As received
1: Swapped MSB vs. LSB

Bit 25 **SWAPCYCLES**: Swap data (cycle 0 vs. cycle 1) for pixels received on two cycles
0: Default
1: Swap active: the data of cycle 1 is used before the data of cycle 0.
The swap must not be activated by software for pixels received in one or three cycles.

Bit 24 Reserved, must be kept at reset value.

Bits 23:16 **FORMAT[7:0]**:
0x1E: YUV422
0x22: RGB565
0x24: RGB888 (= YUV444)
0x2A: RAW8
0x2B: RAW10
0x2C: RAW12
0x2D: RAW14
0x4A: monochrome 8-bit
0x4B: monochrome 10-bit
0x4C: monochrome 12-bit
0x4D: monochrome 14-bit
0x5A: byte stream (JPEG, compressed video)
Other values: data are captured and output as-is only through the data/dump pipeline (for example JPEG or byte input format).
The monochrome Y input is inserted in the pipe as YUV pixels, with the U and V components set to neutral, to represent a gray color.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** Parallel interface enable
0: Parallel interface disabled to lower power consumption
1: Parallel interface enabled
The parallel interface configuration registers must be correctly programmed before enabling this bit.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **EDM[2:0]:** Extended data mode
0x0: Interface captures 8-bit data on every pixel clock
0x1: Interface captures 10-bit data on every pixel clock
0x2: Interface captures 12-bit data on every pixel clock
0x3: Interface captures 14-bit data on every pixel clock
0x4: Interface captures 16-bit data on every pixel clock
Other values: Reserved

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **VSPOL:** Vertical synchronization polarity
This bit indicates the level on the VSYNC pin when the data are not valid on the parallel interface.
0: VSYNC active low
1: VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity
This bit indicates the level on the HSYNC pin when the data are not valid on the parallel interface.
0: HSYNC active low
1: HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity
This bit configures the capture edge of the pixel clock
0: Falling edge active
1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the HSYNC/VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when this bit is set.

Bits 3:0 Reserved, must be kept at reset value.

42.14.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESCR)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEC[7:0]								LEC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSC[7:0]								FSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEC[7:0]:** Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF00 00XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]:** Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]:** Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]:** Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected, but the first occurrence of LSC after an FEC code is interpreted as the start of frame delimiter.

42.14.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR)

Address offset: 0x10C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEU[7:0]								LEU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSU[7:0]								FSU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEU[7:0]:** Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMIPP_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMIPP_ESCR is compared while comparing the frame end delimiter with the received data

Bits 23:16 **LEU[7:0]:** Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMIPP_ESCR is masked while comparing the line end delimiter with the received data

1: The corresponding bit in the LEC byte in DCMIPP_ESCR is compared while comparing the line end delimiter with the received data

Bits 15:8 **LSU[7:0]:** Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMIPP_ESCR is masked while comparing the line start delimiter with the received data

1: The corresponding bit in the LSC byte in DCMIPP_ESCR is compared while comparing the line start delimiter with the received data

Bits 7:0 **FSU[7:0]:** Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.

0: The corresponding bit in the FSC byte in DCMIPP_ESCR is masked while comparing the frame start delimiter with the received data

1: The corresponding bit in the FSC byte in DCMIPP_ESCR is compared while comparing the frame start delimiter with the received data

42.14.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER)

Address offset: 0x1F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	Res.	Res.	Res.	Res.	Res.	Res.
									rw						

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **ERRIE:** Synchronization error interrupt enable

0: No interrupt generation

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

42.14.12 DCMIPP parallel interface status register (DCMIPP_PRSR)

Address offset: 0x1F8

Reset value: 0x0003 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSYNC	HSYNC
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	Res.	Res.	Res.	Res.	Res.	Res.
									r						

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VSYNC:**

This bit gives the state of the VSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit, and cleared otherwise.

When embedded synchronization codes are used:

0: Active frame

1: Synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

Bit 16 **HSYNC:**

This bit gives the state of the HSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit, and cleared otherwise.

When embedded synchronization codes are used:

0: Active line

1: Synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **ERRF:** Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CERRF bit in DCMIPP_PRFCR.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

42.14.13 DCMIPP parallel interface interrupt clear register (DCMIPP_PRFCR)

Address offset: 0x1FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	Res.	Res.	Res.	Res.	Res.	Res.
									w						

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CERRF:** Synchronization error interrupt status clear

Writing a 1 into this bit clears the ERRF bit in DCMIPP_PRSR.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

42.14.14 DCMIPP common IP HW configuration register (DCMIPP_CMHWCFGR)

Address offset: 0x200

Reset value: 0x0000 007C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPG	CCSI	CPAR	CPIPES[1:0]		Res.	Res.
									r	r	r	r	r		

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TPG:** Presence of the test pattern generator
0: TPG is not implemented.
1: TPG is implemented.

Bit 5 **CCSI:** Presence of the CSI-2 host interface
0: No input from CSI-2 host.
1: Input from CSI-2 host is implemented (independently from the presence of CSI-2 host).

Bit 4 **CPAR:** Presence of the parallel interface
0: Parallel interface input is not implemented.
1: Parallel interface input is implemented.

Bits 3:2 **CPIPES[1:0]:** Amount of instantiated pipes (either dump or pixel pipes)

Bits 1:0 Reserved, must be kept at reset value.

42.14.15 DCMIPP common configuration register (DCMIPP_CMCR)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	Res.	Res.	CFC	Res.	PSFC[1:0]		INSEL
								rw			w		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SWAPRB**: Swap R/U and B/V
0: RGB/VYU
1: BGR/UYV
This bit is not applicable for Pipe0.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CFC**: Clear frame counter

When this bit is set, the frame counter associated to a pipe is cleared. It resets DCMIPP_CMFRCR register. This bit is always read at 0.

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **PSFC[1:0]**: Pipe selection for the frame counter

These bits (set and cleared by software) select on which pipe the frame counter is mapped.
00: Frame counter mapped to Pipe0
01: Frame counter mapped to Pipe1
10: Frame counter mapped to Pipe2
Other values: Reserved

Bit 0 **INSEL**: input selection

0: DCMI
1: CSI-2
This selection is common for all the pipes, meaning that the input source (DCMI or CSI) is exclusive. This bit must not be changed if one or more pipes are enabled or still in processing state after a disabling operation (corresponding CPTACT bit set into the DCMIPP_CMSR1 register).

42.14.16 DCMIPP common frame counter register (DCMIPP_CMFRCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRMCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRMCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **FRMCNT[31:0]**: Frame counter, read-only, loops around.

Incremented following VSYNC detection mapped to the pipe configured into bits PSFC[1:0] of the DCMIPP_CMCR register. The counter is cleared using clear frame counter (CFC) bit of DCMIPP_CMCR register.

42.14.17 DCMIPP common test pattern generator configuration register 1 (DCMIPP_CMTPGCR1)

Address offset: 0x210

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HEIGHT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WIDTH[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **HEIGHT[13:0]**: Height of the visible pixels of the generated frame
Maximum value is 16383, the minimum must comply with the limitations of the DCMIPP.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **WIDTH[13:0]**: Width of the visible pixels of the generated frame
Maximum value is 16383, the minimum is 16.

42.14.18 DCMIPP common test pattern generator configuration register 2 (DCMIPP_CMTPGCR2)

Address offset: 0x214

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VBL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FORMAT[7:0]								RT[1:0]		YT[1:0]		PATTERN	GSEN	LFLEN	TPGEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **VBL[15:0]**: Amount of lines of the vertical blanking
Minimum value is 16.

Bits 15:8 **FORMAT[7:0]**: Pixel format generated:
0x1E: YUV422 (when selected, the YT field defines the YUV subformats)
0x22: RGB565
0x24: RGB888
0x2A: RAW8 (when selected, or RAW10/12/14, RT[1:0] gives the component sequence)
0x2B: RAW10
0x2C: RAW12
0x2D: RAW14
0x4A: monochrome 8-bit
0x4B: monochrome 10-bit
0x4C: monochrome 12-bit
0x4D: monochrome 14-bit
Other values: reserved

Bits 7:6 **RT[1:0]**: Raw Bayer type
Indicates the sequence of the R, G, G, B components. Taken in account when FORMAT is 0x2B, 0x2C, 0x2D:
0x0: RGGB
0x1: GRBG
0x2: GBRG
0x3: BGGR

Bits 5:4 **YT[1:0]**: YUV type
Indicates whether the YUV is defined as CCIR601 or 709, and whether its dynamic is full (0 to 255) or reduced (16 to 240) range.
0x0: BT.601 full range
0x1: BT.709 full range (standard for HDTV)
0x2: BT.601 reduced range (standard for SDTV)
0x3: BT.709 reduced range

Bit 3 **PATTERN**: Pattern selection, between color bars and color squares
0: color squares, with the frame built of adjacent 256 × 256 color squares, each with a soft color gradient across each pixel. Most of the possible colors are present.
1: color bars, with the frame divided vertically into 8 bars of the respective colors: white, yellow, cyan, green, pink, red, blue, black.

Bit 2 **GSEN**: Gray scale enable
0: disabled, with differentiated RGB colors
1: enabled, colors are converted into gray scales

Bit 1 **LFLEN**: Lifeline enable
0: disabled, the lifeline is not visible.
1: enabled, the lifeline is both visible and progressing down of one line per frame

Bit 0 **TPGEN**: Test pattern generator enable
0: disabled
1: enabled, frames are generated continuously, indefinitely

42.14.19 DCMIPP common interrupt enable register (DCMIPP_CMIER)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2 OVR IE	Res.	Res.	Res.	Res.	P2 VSYNC IE	P2 FRAME IE	P2 LINE IE	P1 OVR IE	STOVR IE	STBC IE	ST FRAME IE	Res.	P1 VSYNC IE	P1 FRAME IE	P1 LINE IE
rw					rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0 OVR IE	P0 LIMIT IE	Res.	Res.	Res.	P0 VSYNC IE	P0 FRAME IE	P0 LINE IE	Res.	PR ERR IE	ATX ERR IE	Res.	Res.	Res.	Res.	Res.
rw	rw				rw	rw	rw		rw	rw					

Bit 31 **P2OVRIE:** Overrun interrupt status enable for Pipe2
0: No interrupt generation
1: An interrupt is generated

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **P2VSYNCIE:** Vertical sync interrupt enable for Pipe2
0: No interrupt generation
1: An interrupt is generated

Bit 25 **P2FRAMEIE:** Frame capture complete interrupt enable for Pipe2
0: No interrupt generation
1: An interrupt is generated

Bit 24 **P2LINEIE:** Multi-line capture complete interrupt enable for Pipe2
0: No interrupt generation
1: An interrupt is generated

Bit 23 **P1OVRIE:** Overrun interrupt enable for Pipe1

Bit 22 **STOVRIE:** Histogram statistics overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received

Bit 21 **STBCIE:** Histogram statistics bad configuration interrupt enable
0: No interrupt generation
1: An interrupt is generated if the software configuration is considered as non consistent

Bit 20 **STFRAMEIE:** Histogram statistics frame capture completed interrupt enable
0: No interrupt generation
1: An interrupt is generated after the full capture of the statistics of a frame

Bit 19 Reserved, must be kept at reset value.

Bit 18 **P1VSYNCIE:** Vertical sync interrupt enable for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 17 **P1FRAMEIE:** Frame capture complete interrupt enable for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 16 **P1LINEIE:** Multi-line capture complete interrupt status clear for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 15 **P0OVRIE:** Overrun interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 14 **P0LIMITIE:** Limit interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **P0VSYNCIE:** Vertical sync interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 9 **P0FRAMEIE:** Frame capture complete interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 8 **P0LINEIE:** Multi-line capture complete interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PRERRIE:** Limit interrupt enable for the parallel Interface
0: No interrupt generation
1: An interrupt is generated

Bit 5 **ATXERRIE:** AXI transfer error interrupt enable for IP-Plug
0: No interrupt generation
1: An interrupt is generated

Bits 4:0 Reserved, must be kept at reset value.

42.14.20 DCMIPP common status register 1 (DCMIPP_CMSR1)

Address offset: 0x3F4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2CPTACT	Res.	Res.	Res.	Res.	Res.	P2LSTFRM	P2LSTLINE	P1CPTACT	Res.	Res.	Res.	Res.	Res.	P1LSTFRM	P1LSTLINE
r						r	r	r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0CPTACT	Res.	Res.	Res.	Res.	Res.	P0LSTFRM	P0LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	PRVSYNC	PRHSYNC
r						r	r							r	r

Bit 31 **P2CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe2
0: No capture currently active
1: Capture currently active

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **P2LSTFRM**: Last frame LSB bit, sampled at frame capture complete event for Pipe2
The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 24 **P2LSTLINE**: Last line LSB bit, sampled at frame capture complete event for Pipe2
The first line of a frame is counted as 0.

Bit 23 **P1CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe1
0: No capture currently active
1: Capture currently active

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **P1LSTFRM**: Last frame LSB bit, sampled at frame capture complete event for Pipe1
The information is extracted from the frame data number, which can be delivered by the camera through the CSI2 interface.

Bit 16 **P1LSTLINE**: Last line LSB bit, sampled at Frame capture complete event for Pipe1

Bit 15 **P0CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe0
0: No capture currently active
1: Capture currently active

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **P0LSTFRM**: Last frame LSB bit, sampled at Frame capture complete event for Pipe0

The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 8 **P0LSTLINE**: Last line LSB bit, sampled at Frame capture complete event for Pipe0

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **PRVSYNC:**

This bit gives the state of the VSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: Active frame

1: Synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

Bit 0 **PRHSYNC:**

This bit gives the state of the HSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used the meaning of this bit is the following:

0: Active line

1: Synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

42.14.21 DCMIPP common status register 2 (DCMIPP_CMSR2)

Address offset: 0x3F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2 OVRF	Res.	Res.	Res.	Res.	P2 VSYNCF	P2 FRAMEF	P2 LINEF	P1 OVRF	ST OVRF	STBCF	ST FRAMEF	Res.	P1 VSYNCF	P1 FRAMEF	P1LINEF
r					r	r	r	r	r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0 OVRF	P0 LIMITF	Res.	Res.	Res.	P0 VSYNCF	P0 FRAMEF	P0 LINEF	Res.	PR ERRF	ATX ERRF	Res.	Res.	Res.	Res.	Res.
r	r				r	r	r		r	r					

Bit 31 **P2OVRF:** Overrun raw interrupt status for Pipe2

0: No data buffer overrun occurred

1: A data buffer overrun occurred, and this frame data are corrupted

This bit is cleared by writing 1 to the CP2OVRF bit in DCMIPP_CMFCR.

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **P2VSYNCF:** VSYNC raw interrupt status for Pipe2

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CP2VSYNCF bit in the DCMIPP_CMFCR register.

Bit 25 **P2FRAMEF:** Frame capture completed raw interrupt status for Pipe2

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CP2FRAMEF bit in the DCMIPP_CMFCR.

Bit 24 **P2LINEF:** Multi-line capture completed raw interrupt status for Pipe2

This bit is set when at least a line has been completed, and cleared by writing 1 to the CP2LINEF bit in the DCMIPP_CMFCR register.

The periodicity of LINEF event is configured by LINEMULT bits of the DCMIPP_P2PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame. For embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set.

Bit 23 **P1OVRF:** Overrun raw interrupt status for Pipe1

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the CP1OVRF bit in the DCMIPP_CMFCR register.

Bit 22 **STOVRF:** Histogram statistics overrun interrupt status

0: No overrun occurred

1: An overrun occurred

Bit 21 **STBCF:** Histogram statistics bad configuration interrupt status

0: No overrun occurred

1: An overrun occurred

Bit 20 **STFRAMEF:** Histogram statistics frame capture completed interrupt enable

0: No overrun occurred

1: An overrun occurred

Bit 19 Reserved, must be kept at reset value.

Bit 18 **P1VSYNCF:** VSYNC raw interrupt status for Pipe1

This bit is set when the VSYNC signal changes from inactive to active state. In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CP1VSYNCF bit in the DCMIPP_CMFCR register.

Bit 17 **P1FRAMEF:** Frame capture completed raw interrupt status for Pipe1

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured. In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame). This bit is cleared by writing 1 to the CP1FRAMEF bit in the DCMIPP_CMFCR register.

Bit 16 **P1LINEF:** Multi-line capture completed raw interrupt status for Pipe1

This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P1PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame. In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMIPP_CR is set. It is cleared by writing 1 to the CP1LINEF bit in the DCMIPP_CMFCR register.

Bit 15 **P0OVRF:** Overrun raw interrupt status for Pipe0

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the CP0OVRF bit in the DCMIPP_CMFCR register.

Bit 14 **P0LIMITF:** Limit raw interrupt status for Pipe0

This bit is set when the data counter DCMIPP_P0DCCNT reaches its maximum value DCMIPP_P0DCLIMIT. It is cleared by writing 1 to the CP0LIMITF bit in the DCMIPP_CMFCR register.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **P0VSYNCF:** VSYNC raw interrupt status for Pipe0

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR. It is cleared by writing 1 to the CP0VSYNCF bit in the DCMIPP_CMFCR register.

Bit 9 **P0FRAMEF:** Frame capture completed raw interrupt status for Pipe0

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop, even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CP0FRAMEF bit in the DCMIPP_CMFCR register.

Bit 8 **P0LINEF:** Multi-line capture completed raw interrupt status for Pipe0

This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P0PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set. It is cleared by writing 1 to the CP0LINEF bit in the DCMIPP_CMFCR register.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PRERRF:** Synchronization error raw interrupt status for the parallel interface.

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CPRERRF bit in the DCMIPP_CMFCR register.

This bit is available only in embedded synchronization mode.

Bit 5 **ATXERRF:** AXI transfer error interrupt status flag for the IP-Plug.

0: No AXI transfer error detected

1: AXI transfer error occurred on an AXI client. This bit signals an error on a client without any specific hardware action, the software must handle the situation (normally used when debugging software application code).

This bit is cleared by writing 1 to CATXERRF bit in the DCMIPP_CMFCR register.

Bits 4:0 Reserved, must be kept at reset value.

42.14.22 DCMIPP common interrupt clear register (DCMIPP_CMFCR)

Address offset: 0x3FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CP2 OVRF	Res.	Res.	Res.	Res.	CP2 VSYNCF	CP2 FRAMEF	CP2 LINEF	CP1 OVRF	CST OVRF	CST BCF	CST FRAMEF	Res.	CP1 VSYNCF	CP1 FRAMEF	CP1 LINEF
w					w	w	w	w	w	w	w		w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP0 OVRF	CP0 LIMITF	Res.	Res.	Res.	CP0 VSYNCF	CP0 FRAMEF	CP0 LINEF	Res.	CPR ERRF	CATX ERRF	Res.	Res.	Res.	Res.	Res.
w	w				w	w	w		w	w					

Bit 31 **CP2OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P2OVRF bit in the DCMIPP_CMSR2 register

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CP2VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P2VSYNCF bit in the DCMIPP_CMSR2 register

Bit 25 **CP2FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P2FRAMEF bit in the DCMIPP_CMSR2 register

Bit 24 **CP2LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P2LINEF in the DCMIPP_CMSR2 register

Bit 23 **CP1OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P1OVRF bit in the DCMIPP_CMSR2 register

Bit 22 **CSTOVRF:** Histogram statistics overrun interrupt clear
Writing 1 into this bit clears the STOVRF bit in the DCMIPP_CMSR2 register.

Bit 21 **CSTBCF:** Histogram statistics bad configuration interrupt enable
Writing 1 into this bit clears the STBCF bit in the DCMIPP_CMSR2 register.

Bit 20 **CSTFRAMEF:** Histogram statistics frame capture completed interrupt enable
Writing 1 into this bit clears the STFRAMEF bit in the DCMIPP_CMSR2 register.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CP1VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P1VSYNCF bit in the DCMIPP_CMSR2 register

Bit 17 **CP1FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P1FRAMEF bit in the DCMIPP_CMSR2 register

Bit 16 **CP1LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P1LINEF in the DCMIPP_CMSR2 register

Bit 15 **CP0OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P0OVRF bit in the DCMIPP_CMSR2 register

Bit 14 **CP0LIMITF:** limit interrupt status clear
Writing 1 into this bit clears P0LIMITF in the DCMIPP_CMSR2 register

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **CP0VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P0VSYNCF bit in the DCMIPP_CMSR2 register.

Bit 9 **CP0FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P0FRAMEF bit in the DCMIPP_CMSR2 register.

Bit 8 **CP0LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P0LINEF in the DCMIPP_CMSR2 register

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CPRERRF:** Synchronization error interrupt status clear
Writing 1 into this bit clears the PRERRF bit in the DCMIPP_CMSR2 register.
This bit is available only in embedded synchronization mode.

Bit 5 **CATXERRF:** AXI transfer error interrupt status clear
Writing 1 into this bit clears the ATXERRF bit in the DCMIPP_CMSR2 register.

Bits 4:0 Reserved, must be kept at reset value.

42.14.23 DCMIPP Pipe0 HW configuration register (DCMIPP_P0HWCFGR)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CROI[3:0]				CGM[1:0]		CRB[1:0]		DBM	CDS	CPLA[1:0]	
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **CROI[3:0]:** Capability for ROI: defines the number of ROIs in the pipe

Bits 7:6 **CGM[1:0]:** Capability for gamma conversions
00: No gamma conversion
01: Gamma conversion with three interpolated segments
10, 11: Reserved.

Bits 5:4 **CRB[1:0]:** Capability for demosaicing
00: No demosaicing
01: Demosaicing with 2018-03 algorithm (Matrix 3 x 3 with Edge/LineHV/Peak)
10, 11: Reserved

Bit 3 **DBM:** Double buffer mode
0: No Double buffer mode supported
1: Double buffer mode supported

Bit 2 **CDS:** Capability for downsize filter
0: No downsize filter, basic decimation instead
1: Downsize filter based on a BoxFilter

Bits 1:0 **CPLA[1:0]:** Capability for planar buffers
00: Capability to dump to a coplanar buffer only
01: Capability to dump to a semi-planar buffer (i.e. YUV420-2)
10: Capability to dump to full-planar buffer (i.e. YUV420-3)
11: Reserved

42.14.24 DCMIPP Pipe0 flow selection configuration register (DCMIPP_P0FSCR)

Address offset: 0x404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC[1:0]		Res.	DTMODE[1:0]	
rw											rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **PIPEN**: Activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Note: This bit is not shadowed, differently from all other bits in this register.

Bits 30:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode
Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode
00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe
01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe
10: All data types from the selected virtual channel (except the DTIDA or DTIDB) are forwarded in the pipe
11: All data types of the selected virtual channel VC are forwarded in the pipe

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Data type selection ID B
Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type selection ID A
Data type ID of the CSI flow to select.

42.14.25 DCMIPP Pipe0 flow control configuration register (DCMIPP_P0FCTCR)

Address offset: 0x500

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested
0: Capture not requested for next frame
1: Capture requested for next frame
When PIPEN = 1 and CPTREQ = 1. the pipe waits for the first VSync, automatically starts a capture, and sets CPTACT = 1 to mention it.
In Snapshot mode the CPTREQ bit is cleared at the start of the first received frame.
In Continuous grab mode, the capture remains active and CPTREQ = 1 until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.
The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode
0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.
00: All frames are captured
01: one out of two frames captured (50% bandwidth reduction)
10: one out of four frames captured (75% bandwidth reduction)
11: one out of eight frames captured (87% bandwidth reduction)

42.14.26 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR)

Address offset: 0x504

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 words wide

42.14.27 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR)

Address offset: 0x508

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:

This bit is set and cleared by software.

0: Bypass. All the data are computed, if the statistic data are sent within the frame, they are sent to the processing pipe as pixels data.

1: Enable. Depending on bit POSNEG value, the rectangle defined by VSIZE, HSIZE, VSTART and HSTART can be used to extract or to remove some data (statistical extraction or removal, or basic 2D crop features).

if POSNEG = 0, the data inside the rectangle area are transmitted (it can correspond to a statistical data removal, or as a crop feature in a data valid image area).

if POSNEG = 1, the data outside of the rectangle area are transmitted (it can correspond to a statistical data extraction, rejecting all data inside the window).

This bit must be kept cleared if the input format is JPEG, to avoid unpredictable behavior of the pipe.

Bit 30 **POSNEG**:

This bit is set and cleared by software. It has a meaning only if ENABLE bit is set.

0: Positive area, the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART

1: Negative area, the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on vertical direction.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 word wide (data 32-bit)
If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on horizontal direction.

42.14.28 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR)

Address offset: 0x5B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CNT[25:16]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **CNT[25:0]**: Number of data dumped during the frame.
The size of the data is expressed in bytes. It counts only the data selected by means of the CROP 2D function. The counter saturates at 0x3FFFFFF. Granularity is 32-bit for all the formats except for the byte stream formats (for example JPEG) having byte granularity.

42.14.29 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR)

Address offset: 0x5B4

Reset value: 0x00FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIMIT[23:16]							
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIMIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:
0: Disabled, no check on the amount of 32-bit words transmitted
1: Enabled, check done versus limit

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:0 **LIMIT[23:0]**: Maximum number of 32-bit data that can be dumped during a frame, after the crop 2D operation.

42.14.30 DCMIPP Pipe0 pixel packer configuration register (DCMIPP_P0PPCR)

Address offset: 0x5C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADER EN	PAD	Res.	Res.	Res.	Res.	SWAP YUV
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw					rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
0x0: Event after one line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS:** Odd/even line select (line select start)
This bit works in conjunction with LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one is dropped
1: Interface captures second line from the frame start, first one is dropped

Bit 10 **LSM:** Line select mode
0: Interface captures all received lines
1: Interface captures one line out of two

Bit 9 **OEBS:** Odd/even byte select (byte select start)
This bit works in conjunction with BSM field (BSM ≠ 00)
0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register.

Bits 8:7 **BSM[1:0]:** Byte select mode
00: Interface captures all received data
01: Interface captures 1 data out of 2
10: Interface captures one byte out of four
11: Interface captures two bytes out of four
Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register

Bit 6 **HEADEREN:** CSI header dump enable
0: CSI-2 headers are not dumped
1: CSI-2 headers are dumped as a 32-bit word.
This setting is ignored for parallel interface mode, as there are no headers.

Bit 5 **PAD:** Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI.
1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **SWAPYUV:** Swaps, within a 32-bit word, byte 0-vs-1 and byte 2-vs-3. It corresponds, for YUV422 pixels formats, to swap between UYVY and YUYV.
0: Outputs the provided words, as described in *Section 42.10: Pixel format description*.
1: Swaps the bytes from provided words, byte 0-vs.-1 and 2-vs.-3

42.14.31 DCMIPP Pipe0 pixel packer Memory0 address register 1 (DCMIPP_P0PPM0AR1)

Address offset: 0x5C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.32 DCMIPP Pipe0 status Memory0 address register (DCMIPP_P0STM0AR)

Address offset: 0x5D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

The register returns the status of the last captured frame.

42.14.33 DCMIPP Pipe0 interrupt enable register (DCMIPP_P0IER)

Address offset: 0x5F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	LIMITIE	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
								rw	rw				rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.

Bit 6 **LIMITIE:** Limit interrupt enable

0: No interrupt generation when the limit is reached

1: An interrupt is generated when the limit is reached

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable

0: No interrupt generation

1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable

0: No interrupt generation when the line is received

1: An interrupt is generated after the full capture of a group of lines (or last line reached)

42.14.34 DCMIPP Pipe0 status register (DCMIPP_P0SR)

Address offset: 0x5F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LST FRM	LST LINE
								r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR F	LIMIT F	Res.	Res.	Res.	VSYNC F	FRAME F	LINE F
								r	r				r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status

0: Capture currently inactive

1: Capture currently active

This bit is automatically reset at the end of frame capture complete event (after all the data of that frame have been captured and the IP-Plug has started to emit the last burst on the AXI, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event.

The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the COVRF bit in the DCMIPP_P0FCR register.

Bit 6 **LIMITF:** Limit raw interrupt status

This bit is set when the data counter DCMIPP_PxDCCNTR reaches its maximum value DCMIPP_PxDCLIMITR.

It is cleared by writing 1 to the CLIMITF bit in the DCMIPP_P0FCR register.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status

This bit is set when the VSYNC signal changes from the inactive state to the active state. In case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR.

It is cleared by writing 1 to the CVSYNCF bit in the DCMIPP_P0FCR register.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured. In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P0FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status

This bit is set when one/more lines have been completed. For the JPEG mode, this bit is raised at the end of the frame.

The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P0PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.

In case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set. It is cleared by writing 1 to the CLINEF bit in the DCMIPP_P0FCR register.

42.14.35 DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR)

Address offset: 0x5FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CLIMITF	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
								w	w				w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear

Writing 1 into this bit clears the OVRF bit in the DCMIPP_P0SR register.

Bit 6 **CLIMITF:** limit interrupt status clear

Writing 1 into this bit clears LIMITF in the DCMIPP_P0SR register.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear

Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P0SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear

Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P0SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear

Writing 1 into this bit clears LINEF in the DCMIPP_P0SR register.

42.14.36 DCMIPP Pipe0 current flow selection configuration register (DCMIPP_P0CFSCR)

Address offset: 0x604

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC[1:0]		Res.	DTMODE[1:0]	
r											r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bits 30:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode
Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode
00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe.
01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe.
10: All Datatypes from the selected virtual channel, except the DTIDA or DTIDB are forwarded into the pipe.
11: All Datatypes of the selected virtual channel VC are forwarded in the pipe.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Current data type selection ID B
Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type selection ID A
Data type ID of the CSI flow to select.

42.14.37 DCMIPP Pipe0 current flow control configuration register (DCMIPP_P0CFCTCR)

Address offset: 0x700

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame.

1: Capture requested for next frame.

When PIPEN = 1, and when the CPTREQ is set to 1 the pipe waits for the first VSync, and automatically starts a capture and sets CPTACT = 1 to mention it.

In Snapshot mode the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: all frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

42.14.38 DCMIPP Pipe0 current statistic/crop start register (DCMIPP_P0CSCSTR)

Address offset: 0x704

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 words wide

42.14.39 DCMIPP Pipe0 current statistic/crop size register (DCMIPP_P0CSCSZR)

Address offset: 0x708

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]											
r	r			r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current value of the ENABLE bit

0: Bypass. All data are computed, if the statistics data are sent within the frame, they are sent to the processing pipe as pixels data.

1: Enable: Depending on bit POSNEG value, the rectangle defined by the VSIZE, HSIZE, VSTART, HSTART can be used to extract or to remove certain amount of data (statistical extraction or removal, or basic 2D crop features)

if POSNEG = 0, the data inside the rectangle area are transmitted (can correspond to a statistical data removal, or as a crop feature in a data valid image area).

if POSNEG = 1, the data outside of the rectangle area are transmitted (can correspond to a statistical data extraction, rejecting all data inside the window)

Bit 30 **POSNEG**: Current value of the POSNEG bit

This bit has a meaning only if ENABLE bit is set.

0: Positive area. The rectangle defined by VSIZE,HSIZE, and VSTART, HSTART is the active area.

1: Negative area. The active area is the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high.

If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 word wide (data 32-bit).

If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

42.14.40 DCMIPP Pipe0 current pixel packer configuration register (DCMIPP_P0CPPCR)

Address offset: 0x7C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADER EN	PAD	Res.	Res.	Res.	Res.	SWAP YUV
r	r	r		r	r	r	r	r	r	r					r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Current amount of capture completed lines for LINE event and interrupt
0x0: Event after every line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS:** Current odd/even line select (ine select start)
This bit works in conjunction with LSM field (LSM = 1)
0: Interface captures the first line after the frame start, the second one is dropped
1: Interface captures the second line from the frame start, the first one is dropped
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bit 10 **LSM:** Current Line select mode
0: Interface captures all received lines
1: Interface captures one line out of two
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bit 9 **OEBS:** Current odd/even byte select (byte select start)
This bit works in conjunction with BSM field (BSM ≠ 00)
0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bits 8:7 **BSM[1:0]:** Current Byte select mode
00: Interface captures all received data
01: Interface captures one data out of two
10: Interface captures one byte out of four
11: Interface captures two bytes out of four
Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register

Bit 6 **HEADEREN:** Current CSI header dump enable
0: CSI-2 headers are not dumped.
1: CSI-2 headers are dumped as 32-bit words.
This setting is ignored for Parallel interface mode, as there are no headers.

Bit 5 **PAD:** Current Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI
1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **SWAPYUV:** *S*waps, within a 32-bit word, byte 0 vs. 1 and byte 2 vs. 3. It corresponds, for YUV422 pixels formats, to swap between UYVY and YUYV.
0: Outputs the provided words, as described in *Section 42.10: Pixel format description*.
1: Swaps the bytes from provided words, byte 0 vs. 1 and 2 vs. 3.

42.14.41 DCMIPP Pipe0 current pixel packer Memory0 address register 1 (DCMIPP_P0CPPM0AR1)

Address offset: 0x7C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of the current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.42 DCMIPP Pipex HW configuration register (DCMIPP_PxHWCFGR)

Address offset: 0x800 + (x - 1) * 0x400, (x = 1 to 2)

Reset value: 0x0001 1056, 0x0000 1044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFOC	CLS	CHS
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CDEC	CROI[3:0]				CGM[1:0]		CRB[1:0]		DBM	CDS	CPLA[1:0]	
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CFOC:** Capability for focus extraction
- 0: Focus extraction not supported
- 1: Focus extraction supported

Bit 17 **CLS:** Capability for lens shading
- 0: Lens shading not supported
- 1: Lens shading supported

Bit 16 **CHS:** Capability to extract histograms
- 0: Histogram extraction not supported
- 1: Histogram extraction supported

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CDEC:** Capability to decimate before downsize
- 0: No decimation supported
- 1: Decimation before downsize supported

Bits 11:8 **CROI[3:0]:** Capability for ROI: defines the number of ROIs in the pipe

Bits 7:6 **CGM[1:0]:** Capability for gamma conversions
00: No gamma conversion
01: Gamma conversion with seven interpolated segments
10, 11: Reserved

Bits 5:4 **CRB[1:0]:** Capability for demosaicing
00: No demosaicing
01: Demosaicing with 2018-03 algorithm (Matrix 3 x 3 with Edge/LineHV/Peak)
10, 11: Reserved

Bit 3 **DBM:** Double buffer mode
0: No Double buffer mode supported
1: Double buffer mode supported

Bit 2 **CDS:** Capability for downsize filter
0: No downsize filter, basic decimation instead
1: Downsize filter based on a BoxFilter

Bits 1:0 **CPLA[1:0]:** Capability for planar buffers
00: Capability to dump to a coplanar buffer only
01: Capability to dump to a semi-planar buffer (i.e. YUV420-2)
10: Capability to dump to full-planar buffer (i.e. YUV420-3)
11: Reserved

42.14.43 DCMIPP Pipe1 flow selection configuration register (DCMIPP_P1FSCR)

Address offset: 0x804

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		PIPEDIFF	DTMODE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **PIPEN**: Activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Bit 30 **FDTFEN**: Force Datatype format enable
0: Disable force Datatype format. The Datatype format used by the pipe is the one defined into DTIDx1: Enable force Datatype format. When the Datatype DTIDx is detected by the pipe, the real pixel format considered by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 Host. It allows to handle, for instance, 3D sensors camera using user data type in bottom-up approach, and to explicitly define the pixel format to be handled by the pipe. Note that the flow is first selected using the DTMODE and DTIDA/B, and then, if selected, its data type may be forced with the to FDTC using FDTFEN = 1.

Bits 29:24 **FDTF[5:0]**: Force Datatype format

Forcing the Datatype format. This Datatype overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 Datatype format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 **PIPEDIFF:** Differentiates Pipe2 from Pipe1

0: Pipe2 receives the same data as Pipe1 (data filtered by Pipe1 configuration for VC and DTIDA/DTIDB and DTMODE mode, as well as FDTF and FDTFEN bit).

1: Pipe1 gets pixels only from VC and DT configured in Pipe1, processed by all blocks belonging to the image processing functional block (demosaicing, auto exposure, statistics, contrast enhancement). Pipe2 gets pixels only from VC/DTID and FDTFEN/FDTF bits configured in Pipe2, processed from Crop. There are no shared functions with Pipe1, fully independent.

This bit must be kept cleared when INSEL = 0 into the DCMIPP_CMCR register.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode

00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe

01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe

Other values: Reserved.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Data type selection ID B

Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type selection ID A

Data type ID of the CSI flow to select.

42.14.44 DCMIPP Pipe1 stat removal configuration register (DCMIPP_P1SRCR)

Address offset: 0x820

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CROP EN	FIRSTLINEDEL[2:0]			LASTLINE[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CROPEN**: Crop line enable

0: No crop (all pixels are fed through)

1: Crop

Bits 14:12 **FIRSTLINEDEL[2:0]**: Amount of first lines to delete when CROPEN = 1

Bits 11:0 **LASTLINE[11:0]**: Amount of following lines to keep when CROPEN = 1.
If LASTLINE = 0 all pixels after FIRSTLINEDEL are fed through.

42.14.45 DCMIPP Pipe1 bad pixel removal control register (DCMIPP_P1BPRCR)

Address offset: 0x824

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH[2:0]			ENABLE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **STRENGTH[2:0]**: Strength (aggressiveness) of the bad pixel detection
0x0: The filter is fairly tolerant: only a few pixels are declared bad, due to their high noise
...
0x7: The filter is very aggressive: many pixels are declared bad, even if only slightly noisy.

Bit 0 **ENABLE**:
0: Bypass: bad pixel removal is not active, all pixels are transmitted through
1: Enable: if bad pixel are detected, they are replaced by corrected pixels
Bad pixel detection must be enabled only for raw Bayer flows, as it corrupts RGB flows.

42.14.46 DCMIPP Pipe1 bad pixel removal status register (DCMIPP_P1BPRSR)

Address offset: 0x828

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BADCNT[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **BADCNT[11:0]**: Amount of detected bad pixels
Counts the amount of bad pixels detected in each frame, and if too many, saturates at its maximum value (4095). It is reclocked at the frame end, to remain stable for reading during the next frame.

42.14.47 DCMIPP Pipe1 decimation register (DCMIPP_P1DECR)

This register controls the input decimation.

Address offset: 0x830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio

0x0: All pixels are transmitted, no vertical decimation

0x1: one line out of two transmitted (for raw Bayer data, two lines of components every four)

0x2: one line out of four transmitted (for raw Bayer data, two lines of components every eight)

0x3: one line out of eight transmitted (for raw Bayer data, two lines of components every 16)

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio

0x0: All pixels are transmitted, no horizontal decimation

0x1: one line out of two transmitted (for raw Bayer data, two lines of components every four)

0x2: one line out of four transmitted (for raw Bayer data, two lines of components every eight)

0x3: one line out of eight transmitted (for raw Bayer data, two lines of components every 16)

Bit 0 **ENABLE**:

0: Bypass: decimation is not active, all pixels are transmitted

1: Enable: one pixel every 1, 2, 4, or 8, in H and V is transmitted, following HDEC and VDEC

42.14.48 DCMIPP Pipe1 black level calibration control register (DCMIPP_P1BLCCR)

Address offset: 0x840

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLCR[7:0]								BLCG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw								rw

Bits 31:24 **BLCR[7:0]**: Black level calibration - Red

Value subtracted from the red input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 23:16 **BLCG[7:0]**: Black level calibration - Green

Value subtracted from the green input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 15:8 **BLCB[7:0]**: Black level calibration - Blue
Value subtracted from the blue input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Black level calibration
0: Bypass: black level calibration is not active, all pixels are transmitted non-modified.
1: Enable: the BLC R, G, B are subtracted.

42.14.49 DCMIPP Pipe1 exposure control register 1 (DCMIPP_P1EXCR1)

Address offset: 0x844

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFR[2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFR[2:0]**: Exposure shift - Red
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of reds is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTR[7:0]**: Exposure multiplier - Red
Value that is multiplied for the red input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the reds) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the reds as-is).
Multiplying by 255 increase the exposure (lightens the reds) by a factor ~2x.

Bits 19:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Exposure control (multiplication and shift) of all red, green and blue
0: Bypass: exposure multiplier and shift are not applied
1: Enable: the exposure multiplication and shift is applied

42.14.50 DCMIPP Pipe1 exposure control register 2 (DCMIPP_P1EXCR2)

Address offset: 0x848

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFG[2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SHFB[2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFG[2:0]**: Exposure shift - Green

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: Exposure of greens is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTG[7:0]**: Exposure multiplier - Green

Value that is multiplied for the green input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the greens) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the greens as-is).
Multiplying by 255 increases the exposure (lightens the greens) by a factor ~2x.

Bits 19:15 Reserved, must be kept at reset value.

Bits 14:12 **SHFB[2:0]**: Exposure shift - Blue

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of blues is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 11:4 **MULTB[7:0]**: Exposure multiplier - Blue

Value that is multiplied for the blue input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the blues) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the blues as-is).
Multiplying by 255 increases the exposure (lightens the blues) by a factor ~2x.

Bits 3:0 Reserved, must be kept at reset value.

42.14.51 DCMIPP Pipe1 statistics1 control register (DCMIPP_P1ST1CR)

Address offset: 0x850

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
Typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
Typically used to extract how many pixel components fall into the various dynamic bins, and to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled after black level, and sample, respectively, red, green, blue, and luminance. For raw Bayer formats, only one component is available by pixel.
0x4, 0x5, 0x6, 0x7: components are sampled after demosaicing, and sample, respectively, red, green, blue, and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: All Pixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enable: statistics are accumulated

42.14.52 DCMIPP Pipe1 statistics 2 control register (DCMIPP_P1ST2CR)

Address offset: 0x854

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode
0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source
0x0, 0x1, 0x2, 0x3: components are sampled after black level, and sample, respectively, the red, green, blue components, and luminance.
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component < 4 (< 8, < 16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component < 32 (< 64, < 128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component > 127 (> 191, > 224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component > 239 (> 247, > 251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240.
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224.
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enabled: statistics are accumulated

42.14.53 DCMIPP Pipe1 statistics 3 control register (DCMIPP_P1ST3CR)

Address offset: 0x858

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode
0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source
0x0, 0x1, 0x2, 0x3: components are sampled after black level, and sample, respectively, the red, green, blue components, and luminance.
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, the red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240.
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224.
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enabled: statistics are accumulated

42.14.54 DCMIPP Pipe1 statistics window start register (DCMIPP_P1STSTR)

Address offset: 0x85C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels wide

42.14.55 DCMIPP Pipe1 statistics window size register (DCMIPP_P1STSZR)

Address offset: 0x860

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CROPEN	Res.	Res.	Res.	VSIZE[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CROPEN**:

0: Bypass, all pixels are used to compute the statistics

1: Enable, only the rectangle defined by VSTART, HSTART, VSIZE, HSIZE is used for statistics.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

If the value is kept at 0 when enabling the crop with the CROPEN bit, the value is forced internally at the maximum value 0xFFE.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide

If the value is kept at 0 when enabling the crop with the CROPEN bit, the value is forced internally at the maximum value 0xFFE.

42.14.56 DCMIPP Pipe1 statistics 1 status register (DCMIPP_P1ST1SR)

Address offset: 0x864

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: Accumulation result, divided by 256.

42.14.57 DCMIPP Pipe1 statistics 2 status register (DCMIPP_P1ST2SR)

Address offset: 0x868

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: accumulation result, divided by 256.

42.14.58 DCMIPP Pipe1 statistics 3 status register (DCMIPP_P1ST3SR)

Address offset: 0x86C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: accumulation result, divided by 256.

42.14.59 DCMIPP Pipe1 demosaicing configuration register (DCMIPP_P1DMCR)

Address offset: 0x870

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EDGE[2:0]			Res.	LINEH[2:0]			Res.	LINEV[2:0]			Res.	PEAK[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE[1:0]		ENABLE
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **EDGE[2:0]**: Strength of the edge detection
0x0: No edge detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **LINEH[2:0]**: Strength of the horizontal line detection
0x0: No horizontal line detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **LINEV[2:0]**: Strength of the vertical line detection
0x0: No vertical line detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PEAK[2:0]**: Strength of the peak detection
0x0: No peak detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TYPE[1:0]**: Raw Bayer type
0x0: RGGB
0x1: GRBG
0x2: GBRG
0x3: BGGR

Bit 0 **ENABLE**:
0: Bypass
1: Enable demosaicing

42.14.60 DCMIPP Pipe1 ColorConv configuration register (DCMIPP_P1CCCR)

Address offset: 0x880

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: output samples type used while CLAMP is activated
0: Clamped to [16;235] for Y and to [16;240] for U and V
1: Clamped to [16;235] for R, G and B

Bit 0 **ENABLE**:
0: ColorConv is bypassed
1: ColorConv is enabled

42.14.61 DCMIPP Pipe1 ColorConv red coefficient register 1 (DCMIPP_P1CCRR1)

Address offset: 0x884

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100). The sign is indicated as complement to 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Coefficient row 1 column 1 of the matrix

42.14.62 DCMIPP Pipe1 ColorConv red coefficient register 2 (DCMIPP_P1CCRR2)

Address offset: 0x888

Reset value: 0x0000 0000

The coefficients RA[9:0] of the added column, for the ColorConv, are defined as signed 10-bit integer with the sign as complement to 2 (-256 is coded by 0x200).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Coefficient row 1 column 3 of the matrix

42.14.63 DCMIPP Pipe1 ColorConv green coefficient register 1 (DCMIPP_P1CCGR1)

Address offset: 0x88C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Coefficient row 2 column 1 of the matrix

42.14.64 DCMIPP Pipe1 ColorConv green coefficient register 2 (DCMIPP_P1CCGR2)

Address offset: 0x890

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Coefficient row 2 column 3 of the matrix

42.14.65 DCMIPP Pipex ColorConv blue coefficient register 1 (DCMIPP_P1CCBR1)

Address offset: 0x894

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Coefficient row 3 column 1 of the matrix

42.14.66 DCMIPP Pipe1 ColorConv blue coefficient register 2 (DCMIPP_P1CCBR2)

Address offset: 0x898

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Coefficient row 3 column 3 of the matrix

42.14.67 DCMIPP Pipe1 contrast control register 1 (DCMIPP_P1CTCR1)

Address offset: 0x8A0

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM0[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	rw	rw	rw	rw	rw	rw									rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM0[5:0]**: Luminance increase for input luminance of 0 (increase is idle with LUMx = 16)

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Bypass, pixels are forwarded idle.

1: Enable, contrast enhancement is applied on pixels.

42.14.68 DCMIPP Pipe1 contrast control register 2 (DCMIPP_P1CTCR2)

Address offset: 0x8A4

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM1[5:0]**: Luminance increase for input luminance of 32 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM2[5:0]**: Luminance increase for input luminance of 64 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM3[5:0]**: Luminance increase for input luminance of 96 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM4[5:0]**: Luminance increase for input luminance of 128 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

42.14.69 DCMIPP Pipe1 contrast control register 3 (DCMIPP_P1CTCR3)

Address offset: 0x8A8

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM5[5:0]**: Luminance increase for input luminance of 160 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM6[5:0]**: Luminance increase for input luminance of 192 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM7[5:0]**: Luminance increase for input luminance of 224 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM8[5:0]**: Luminance increase for input luminance of 256 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

42.14.70 DCMIPP Pipe1 histogram configuration register (DCMIPP_P1HSCR)

Address offset: 0x8B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	BIN[1:0]		DYN[1:0]		Res.	COMP[2:0]			Res.	VDEC[2:0]		
				rw	rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HDEC[2:0]			VREG[3:0]				HREG[3:0]				SRC[2:0]			EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:26 **BIN[1:0]**: Bin amount per histogram. All bins have the same width.
0x0: 4 bins per histogram (assuming an input with 256 values, each bin is 64-value wide)
0x1: 16 bins per histogram (assuming an input with 256 values, each bin is 16-value wide)
0x2: 64 bins per histogram (assuming an input with 256 values, each bin is 4-value wide)
0x3: 256 bins per histogram (assuming an input with 256 values, each bin is 1-value wide)

Bits 25:24 **DYN[1:0]**: Dynamic of the pixel value, when extracting the histogram

The input pixel value is assumed mapped on 14 bits, 8 are extracted from them to select the bin to update. It is possible to map non-msb bits, to focus the histogram dynamic on the darker pixels. When non-msb bits are extracted to select a bin, if the pixel value is larger than the highest bin, then the highest bin is incremented. In other words, the highest bin collects pixel values higher than the selected dynamic.

0x0: pixel[13:6], used to extract a histogram of the light pixel values (default at reset)

0x1: pixel[11:4], used to extract a histogram of the semi light pixel values

0x2: pixel[9:2], used to extract a histogram of the semi dark pixel values

0x3: pixel[7:0], used to extract a histogram of the dark pixel values

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **COMP[2:0]**: Component selected for the histograms:

Depending on the input source, respectively rawBayer, RGB, YUV, monochrome, the following components are sampled:

0x0: R, R, V, 128

0x1: Gr, G, Y, Y

0x2: B, B, U, 128

0x3: Gb, L, L, L

0x4: All color components are sampled

Other values: reserved.

Note: Gr means the green component among a green-red raw Bayer line, Gb means the green component among a green-blue raw Bayer line. The L component is the luminance, computed as the weighted average of components 0, 1, 2, with 1/4,1/2,1/4 weight. Typically L = R/4 + G/2 +B/4.

Note: If the input pixel format is YUV or monochrome, the four sampled histogram components are filled with, respectively, (V, Y, U, Y) or (128,L, 128,L).

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **VDEC[2:0]**: Vertical decimation ratio

0x0: All pixels are transmitted, no vertical decimation

0x1: one line out of 2 transmitted (if raw Bayer: 2 lines out of 4)

0x2: one line out of 4 transmitted (if raw Bayer: 2 lines out of 8)

0x3: one line out of 8 transmitted (if raw Bayer, 2 lines out of 16)

0x4: one line out of 16 transmitted (if raw Bayer, 2 lines out of 32)

Other values: reserved

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HDEC[2:0]**: Horizontal decimation ratio

0x0: All pixels are transmitted, no vertical decimation

0x1: one line out of 2 transmitted (if raw Bayer: 2 lines out of 4)

0x2: one line out of 4 transmitted (if raw Bayer: 2 lines out of 8)

0x3: one line out of 8 transmitted (if raw Bayer, 2 lines out of 16)

0x4: one line out of 16 transmitted (if raw Bayer, 2 lines out of 32)

Other values: reserved

Bits 11:8 **VREG[3:0]**: Amount of vertical regions per frame, from which to extract the histograms:

0x0: 1 region per frame

0x1: 2 regions per frame

...

0xF: 16 regions per frame

Bits 7:4 **HREG[3:0]**: Amount of horizontal regions per frame, from which to extract the histograms
0x0: 1 region per frame
0x1: 2 regions per frame
...
0xF: 16 regions per frame

Bits 3:1 **SRC[2:0]**: Source across the pipeline, where to sample pixels and extract the histograms:
0x0: after the input decimation (and before the black level)
0x1: after the black level (and before the exposure)
0x2: after the exposure (and before the demosaicing)
0x3: after the demosaicing (and before the ColorConv)
0x4: after the ColorConv (and before the contrast)
0x5: after the contrast (and before entering the applicative pipe and its CROP)
Other values: reserved

Note: When SRC > 2 the pixel is considered RGB.

Bit 0 **EN**: Histogram enable
0: disabled
1: enabled, histograms are extracted and written out.

42.14.71 DCMIPP Pipe1 histogram window start register (DCMIPP_P1HSSTR)

Address offset: 0x8B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high.
The unit is the pixel. In case of a raw Bayer input, VSTART is rounded to even values.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels high.
TThe unit is the pixel. In case of a raw Bayer input, HSTART is rounded to even values.

42.14.72 DCMIPP Pipe1 histogram window size register (DCMIPP_P1HSSZR)

Address offset: 0x8B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size of each histogram region, from 0 to 4094 pixels high

The unit is the pixel. It must be a multiple of decimation. In case of a raw Bayer input, VSIZE is rounded to even values.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size of each histogram region, from 0 to 4094 pixels high

The unit is the pixel. It must be a multiple of decimation. In case of a raw Bayer input, HSIZE is rounded to even values.

42.14.73 DCMIPP Pipe1 histogram memory address register 1 (DCMIPP_P1HSMAR1)

Address offset: 0x8BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **MA[31:0]**: Memory address

Base address of memory area, to whom histogram data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.74 DCMIPP Pipe1 histogram memory status address register 3 (DCMIPP_P1HSSMAR)

Address offset: 0x8C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MA[31:0]**: Memory address

Base address of memory area, to whom histogram data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

This register returns the status of the last captured histogram.

42.14.75 DCMIPP Pipex flow control configuration register (DCMIPP_PxFCTCR)

Address offset: 0x900 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame

1: Capture requested for next frame

When PIPEN = 1, and when the CPTREQ is set to 1, the pipe waits for the first VSync, then automatically starts a capture and sets CPTACT = 1 to mention it.

In Snapshot mode, the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode, the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode. The received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame). Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: All frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

42.14.76 DCMIPP Pipex crop window start register (DCMIPP_PxCRSTR)

Address offset: 0x904 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels wide

42.14.77 DCMIPP Pipex crop window size register (DCMIPP_PxCRSZR)

Address offset: 0x908 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	VSIZE[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:

0: Bypass, all pixels are transmitted through (no need to set any start and size values)

1: Enable, only the rectangle defined by VSIZE, HSIZE, VSTART, HSTART is transmitted.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high. If the value is maintained at 0 when enabling the crop thanks to the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide. If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

42.14.78 DCMIPP Pipex decimation register (DCMIPP_PxDCCR)

This register controls decimation, prior to downsize (to complement it).

Address offset: 0x90C + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio
- 0x0: all pixels are transmitted, no vertical decimation
- 0x1: one line out of two transmitted
- 0x2: one line out of four transmitted
- 0x3: one line out of eight transmitted

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio
- 0x0: all pixels are transmitted, no horizontal decimation
- 0x1: one line out of two transmitted
- 0x2: one line out of four transmitted
- 0x3: one line out of eight transmitted

Bit 0 **ENABLE**:
- 0: Bypass: decimation is not active, all pixels are transmitted
- 1: Enable: one pixel every 1, 2, 4 or 8, in H and V is transmitted, following HDEC and VDEC

42.14.79 DCMIPP Pipex downsize configuration register (DCMIPP_PxDSCR)

Address offset: 0x910 + 0x400 * (x - 1) (x = 1 to 2)

Reset value: 0x0000 0000

The division factor is an unsigned integer based on the source/destination ratio:

div = max(int(1024 / (source/destination)), 1023)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]									
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:
0: Down scaler is bypassed
1: Down scaler is enabled

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:16 **VDIV[9:0]**: Vertical division factor, from 128 (8x) to 1023 (1x)

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **HDIV[9:0]**: Horizontal division factor, from 128 (8x) to 1023 (1x)

42.14.80 DCMIPP Pipex downsize ratio register (DCMIPP_PxDSRTIOR)

Address offset: 0x914 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The down scale ratio is expressed in unsigned 3.13 format:

ratio = max(int((SourceSize - 1) / (DestinationSize -1)) * 8192**)**, 8 * 8192 - 1**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VRATIO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRATIO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **VRATIO[15:0]**: Vertical ratio, from 8192 (1x) to 65535 (8x)

Bits 15:0 **HRATIO[15:0]**: Horizontal ratio, from 8192 (1x) to 65535 (8x)

42.14.81 DCMIPP Pipex downsize destination size register (DCMIPP_PxDSSZR)

Address offset: 0x918 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide

42.14.82 DCMIPP Pipex gamma configuration register (DCMIPP_PxGMCR)

Address offset: 0x970 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Gamma is bypassed
1: Gamma is enabled

42.14.83 DCMIPP Pipe1 YUVConv configuration register (DCMIPP_P1YUVCR)

Address offset: 0x980

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: Output samples type used while CLAMP is activated
0: Clamped to [16;235] for Y and to [16;240] for U and V
1: Clamped to [16;235] for R, G and B

Bit 0 **ENABLE**:
0: ColorConv is bypassed
1: ColorConv is enabled

42.14.84 DCMIPP Pipe1 YUVConv red coefficient register 1 (DCMIPP_P1YUVRR1)

Address offset: 0x984

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100):

int_value = int(float_value * (1« 8) + 0.5**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Coefficient row 1 column 1 of the matrix

42.14.85 DCMIPP Pipe1 YUVConv red coefficient register 2 (DCMIPP_P1YUVRR2)

Address offset: 0x988

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Coefficient row 1 column 3 of the matrix

42.14.86 DCMIPP Pipe1 YUVConv green coefficient register 1 (DCMIPP_P1YUVGR1)

Address offset: 0x98C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Coefficient row 2 column 1 of the matrix

42.14.87 DCMIPP Pipe1 YUVConv green coefficient register 2 (DCMIPP_P1YUVGR2)

Address offset: 0x990

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Coefficient row 2 column 3 of the matrix

42.14.88 DCMIPP Pipe1 YUVConv blue coefficient register 1 (DCMIPP_P1YUVBR1)

Address offset: 0x994

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Coefficient row 3 column 1 of the matrix

42.14.89 DCMIPP Pipe1 YUV blue coefficient register 2 (DCMIPP_P1YUVBR2)

Address offset: 0x998

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Coefficient row 3 column 3 of the matrix

42.14.90 DCMIPP Pipe1 pixel packer configuration register (DCMIPP_P1PPCR)

Address offset: 0x9C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
rw	rw	rw									rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE Event and Interrupt
- 0x0: Event after one line
- 0x1: Event after two lines
- 0x2: Event after four lines
- 0x3: Event after eight lines
- 0x4: Event after sixteen lines
- 0x5: Event after 32 lines
- 0x6: Event after 64 lines
- 0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and U-vs-V components if YUV
0: No swap of R-vs-B (U-vs-V)
1: Swap active

Bits 3:0 **FORMAT[3:0]**: Memory format
0x0: RGB888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888 (with A = 0xFF)
0x3: RGBA8888 (with A = 0xFF)
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (32 bpp, FOURCC = AYUV, with A = 0xFF)
0x6: YUV422 1-buffer (16 bpp, FOURCC = YUYV)
0x7: YUV422 2-buffer (16 bpp, FOURCC = none)
0x8: YUV420 2-buffer (12 bpp, FOURCC = NV21), NV12 available with SWAPRB = 1
0x9: YUV420 3-buffer (12 bpp, FOURCC = YV12)
0xA: YUV422 1-buffer (16 bpp, FOURCC = UYVY)
Other formats are reserved and not supported.

42.14.91 DCMIPP Pipe1 pixel packer Memory0 address register 1 (DCMIPP_P1PPM0AR1)

Address offset: 0x9C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.92 DCMIPP Pipex pixel packer Memory0 pitch register (DCMIPP_PxPPM0PR)

Address offset: 0x9CC + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Number of bytes between the address of two consecutive lines.
It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

42.14.93 DCMIPP Pipex status Memory0 address register (DCMIPP_PxSTM0AR)

Address offset: 0x9D0 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

42.14.94 DCMIPP Pipex pixel packer Memory1 address register 1 (DCMIPP_PxPPM1AR1)

Address offset: 0x9D4 + 0x200 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address
Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.95 DCMIPP Pipex pixel packer Memory1 pitch register (DCMIPP_PxPPM1PR)

Address offset: 0x9DC + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Number of bytes between the address of two consecutive lines.

It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture. PITCH is used also for Memory2 (there is no need to differentiate the pitch of Memory1/Memory2).

42.14.96 DCMIPP Pipex status Memory1 address register (DCMIPP_PxSTM1AR)

Address offset: 0x9E0 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

42.14.97 DCMIPP Pipex pixel packer memory2 address register 1 (DCMIPP_PxPPM2AR1)

Address offset: 0x9E4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of memory area 2 to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.98 DCMIPP Pipex status Memory2 address register (DCMIPP_PxSTM2AR)

Address offset: 0x9F0 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory2 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

42.14.99 DCMIPP Pipe1 interrupt enable register (DCMIPP_P1IER)

Address offset: 0x9F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR IE	STOVR IE	STBC IE	STFRAME IE	Res.	VSYNC IE	FRAME IE	LINE IE
								rw	rw	rw	rw		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.

Bit 6 **STOVRIE:** Statistic overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.

Bit 5 **STBCIE:** Statistic bad configuration interrupt enable
0: No interrupt generation
1: An interrupt is generated if the software configuration is considered inconsistent

Bit 4 **STFRAMEIE:** Statistic frame capture interrupt enable
0: No interrupt generation
1: An interrupt is generated after the full capture of the statistics of a frame

Bit 3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable
0: No interrupt generation
1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable
0: No interrupt generation when the line is received
1: An interrupt is generated after the full capture of a group of lines (or last line reached)

42.14.100 DCMIPP Pipe1 status register (DCMIPP_P1SR)

Address offset: 0x9F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE
								r						r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	STOVRF	STBCF	STFRAMEF	Res.	VSYNCF	FRAMEF	LINEF
								r	r	r	r		r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status
0: Capture currently inactive
1: Capture currently active
This bit is automatically reset at the end of Frame capture complete event (after all the data of that frame have been captured, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event. The information is extracted from the frame data number, which can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status
0: No data buffer overrun occurred
1: A data buffer overrun occurred and this frame data are corrupted.
This bit is cleared by writing 1 to the COVRF bit in DCMIPP_P1FCR.

Bit 6 **STOVRF:** Histogram statistics overrun interrupt status
0: No interrupt occurred
1: An interrupt occurred

Bit 5 **STBCF:** Histogram statistics bad configuration interrupt status
0: No interrupt occurred
1: An interrupt occurred

Bit 4 **STFRAMEF:** Histogram statistics frame capture interrupt status
0: No interrupt occurred
1: An interrupt occurred

Bit 3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status
This bit is set when the VSYNC signal changes from the inactive state to the active state.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR.
It is cleared by writing 1 to the CVSYNCF bit in DCMIPP_P1FCR.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status
0: no capture or ongoing capture
1: all data of a frame have been captured
This bit is set when all data of a frame or window have been captured.
In case of a cropped window, this bit is set at the end of line of the last line in the crop, even if the captured frame is empty (for example window cropped outside the frame).
This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P1FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status
This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P1PPCR. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMIPP_CR is set. It is cleared by writing 1 to the CLINEF bit in DCMIPP_P1FCR.

42.14.101 DCMIPP Pipe1 interrupt clear register (DCMIPP_P1FCR)

Address offset: 0x9FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CSTOVRF	CSTBCF	CSTFRAMEF	Res.	CVSYNCF	CFRAMEF	CLINEF
								w	w	w	w		w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the OVRF bit in the DCMIPP_P1SR register.

Bit 6 **CSTOVRF:** Histogram statistics overrun interrupt clear
Writing 1 into this bit clears the STOVRF bit in the DCMIPP_P1SR register.

Bit 5 **CSTBCF:** Histogram statistics bad configuration interrupt clear
Writing 1 into this bit clears the STBCF bit in the DCMIPP_P1SR register.

Bit 4 **CSTFRAMEF:** Histogram statistics frame capture interrupt clear
Writing 1 into this bit clears the STFRAMEF bit in the DCMIPP_P1SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P1SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P1SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears LINEF in the DCMIPP_P1SR register.

42.14.102 DCMIPP Pipe1 current flow selection configuration register (DCMIPP_P1CFSCR)

Address offset: 0xA04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		PIPEDIFF	DTMODE[1:0]	
r	r	r	r	r	r	r	r				r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN

0: Pipe disabled

1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bit 30 **FDTFEN**: Current force data type format enable

0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx

1: Enable force data type format. When the data type DTIDx is detected by the pipe, the real pixel format considering by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host. It allows to handle, for instance, 3D sensors camera with user data type in bottom-up approach, and to explicitly define the pixel format to be handled by the pipe.

Bits 29:24 **FDTF[5:0]**: Current force data type format

Forcing the data type format. This data type overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 **PIPEDIFF:** Current differentiates Pipe2 vs. Pipe1

0: Pipe2 gets Pipe1 pixels after their processing in shared Gamma, StatRemoval, Demosaicing. Pipe2 shares the same VC DTID of Pipe1 when a CSI2 camera sensor is plugged.

1: Pipe1 gets pixels from only VC/DTID configured in Pipe1, processed by all including Gamma, StatRemoval, Demosaicing. Pipe2 gets pixels from only VC/DTID configured in Pipe2, processed from Crop only (Gamma common with Pipe1, no StatRem, no demosaicing)

Bits 17:16 **DTMODE[1:0]**: Flow selection mode

00: Only flow DTID A from the selected virtual channel is forwarded in the pipe.

01: Flows DTIDA and DTIDB from the selected virtual channel are forwarded in the pipe. This combination is valid only if the FDTFEN bit is set.

Other values: Reserved.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Current data type ID B

Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type ID A

Data type ID of the CSI flow to select.

42.14.103 DCMIPP Pipe1 current bad pixel removal register (DCMIPP_P1CBPRCR)

Address offset: 0xA24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH[2:0]			ENABLE
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **STRENGTH[2:0]**: Current strength (aggressiveness) of the bad pixel detection:
0x0: the filter is fairly tolerant: only a few pixels are declared bad, due to their high noise.
0x7: the filter is very aggressive: many pixels are declared bad, even if only slightly noisy.

Bit 0 **ENABLE**: Current status of enable bit
0: Bypass: bad pixel removal is not active, all pixels are transmitted through
1: Enable: if bad pixel are detected, they are replaced by corrected pixels.
(bad pixel detection must be enabled only for raw Bayer flows, as it would corrupt RGB flows)

42.14.104 DCMIPP Pipe1 current black level calibration control register (DCMIPP_P1CBLCCR)

Address offset: 0xA40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLCR[7:0]								BLCG[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
r	r	r	r	r	r	r	r								r

Bits 31:24 **BLCR[7:0]**: Current black level calibration - Red
Value that is subtracted from the red input components, whether it is on 8,10,12,14 bits.

Bits 23:16 **BLCG[7:0]**: Current black level calibration - Green
Value that is subtracted from the green input components, whether it is on 8,10,12,14 bits.

Bits 15:8 **BLCB[7:0]**: Current black level calibration - Blue
Value that is subtracted from the blue input components, whether it is on 8,10,12,14 bits.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: For current black level calibration
0: Bypass: black level calibration is not active, all pixels are transmitted non-modified.
1: Enable: the BLCR, G, B are subtracted.

42.14.105 DCMIPP Pipe1 current exposure control register 1 (DCMIPP_P1CEXCR1)

Address offset: 0xA44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFR[2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFR[2:0]**: Current exposure shift - Red
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of reds is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTR[7:0]**: Current exposure multiplier - Red
Value that is multiplied for the red input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the reds) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the reds as-is).
Multiplying by 255 increases the exposure (lightens the reds) by a factor ~2x.

Bits 19:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: for exposure control (multiplication and shift)
0: Bypass: exposure multiplier and shift are not applied.
1: Enable: the exposure multiplication and shift is applied.

42.14.106 DCMIPP Pipe1 current exposure control register 2 (DCMIPP_P1CEXCR2)

Address offset: 0xA48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFG[2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SHFB[2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFG[2:0]**: Current exposure shift - Green
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of greens is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTG[7:0]**: Current exposure multiplier - Green
Value that is multiplied for the green input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the greens) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the greens as-is).
Multiplying by 255 increases the exposure (lightens the greens) by a factor ~2x.

Bits 19:15 Reserved, must be kept at reset value.

Bits 14:12 **SHFB[2:0]**: Current exposure shift - Blue

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of blues is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 11:4 **MULTB[7:0]**: Current exposure multiplier - Blue

Value that is multiplied for the blue input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the blues) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the blues as-is).
Multiplying by 255 increases the exposure (lightens the blues) by a factor ~2x.

Bits 3:0 Reserved, must be kept at reset value.

42.14.107 DCMIPP Pipe1 current statistics 1 control register (DCMIPP_P1CST1CR)

Address offset: 0xA50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Current accumulation result, divided by 256.

Bit 7 **MODE**: Current statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Current source of statistics

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition

Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Current enable bit value

0: Disabled: statistics are not accumulated

1: Enable: statistics are accumulated

42.14.108 DCMIPP Pipe1 current statistics 2 control register (DCMIPP_P1CST2CR)

Address offset: 0xA54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Accumulation result, divided by 256.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.

1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins. It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).

0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, the red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition

Condition: MODE = Bins

0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)

0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)

0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)

0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)

Condition: MODE = Average

0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256

0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240

0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224

0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Disabled: statistics are not accumulated

1: Enable: statistics are accumulated

42.14.109 DCMIPP Pipe1 current statistics 3 control register (DCMIPP_P1CST3CR)

Address offset: 0xA58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Accumulation result, divided by 256.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins. It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition

Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Disabled: statistics are not accumulated
1: Enable: statistics are accumulated

42.14.110 DCMIPP Pipe1 current statistics window start register (DCMIPP_P1CSTSTR)

Address offset: 0xA5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 pixels wide

42.14.111 DCMIPP Pipe1 current statistics window size register (DCMIPP_P1CSTSZR)

Address offset: 0xA60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CROPEN	Res.	Res.	Res.	VSIZE[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **CROPEN**: Current CROPEN bit value

0: Bypass, all pixels are used to compute the statistics.

1: Enable, only the rectangle defined by VSTART, HSTART, VSIZE, HSIZE, are used for statistics.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

If the value is maintained at 0 when enabling the crop by means of the CROPEN bit, the value is forced internally at 0xFFE, which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

If the value is maintained at 0 when enabling the crop by means of the CROPEN bit, the value is forced internally at 0xFFE, which is the maximum value.

42.14.112 DCMIPP Pipe1 current ColorConv configuration register (DCMIPP_P1CCCCR)

Address offset: 0xA80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
This bit indicates the current value applied.
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: Output samples type used while CLAMP is activated
This bit indicates the current value applied
0: clamped to [16;235] for Y and to [16;240] for U and V
1: clamped to [16;235] for R, G and B

Bit 0 **ENABLE**: Current value applied
0: ColorConv is bypassed
1: ColorConv is enabled

42.14.113 DCMIPP Pipe1 current ColorConv red coefficient register 1 (DCMIPP_P1CCCRR1)

Address offset: 0xA84

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100):

int_value = int(float_value * (1 « 8) + 0.5**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Current coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Current coefficient row 1 column 1 of the matrix

42.14.114 DCMIPP Pipe1 current ColorConv red coefficient register 2 (DCMIPP_P1CCCRR2)

Address offset: 0xA88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Current coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Current coefficient row 1 column 3 of the matrix

42.14.115 DCMIPP Pipe1 current ColorConv green coefficient register 1 (DCMIPP_P1CCCGR1)

Address offset: 0xA8C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Current coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Current coefficient row 2 column 1 of the matrix

42.14.116 DCMIPP Pipe1 current ColorConv green coefficient register 2 (DCMIPP_P1CCCGR2)

Address offset: 0xA90

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Current coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Current coefficient row 2 column 3 of the matrix

42.14.117 DCMIPP Pipex current ColorConv blue coefficient register 1 (DCMIPP_P1CCCBR1)

Address offset: 0xA94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Current coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Current coefficient row 3 column 1 of the matrix

42.14.118 DCMIPP Pipe1 current ColorConv blue coefficient register 2 (DCMIPP_P1CCCBR2)

Address offset: 0xA98

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Current coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Current coefficient row 3 column 3 of the matrix

42.14.119 DCMIPP Pipe1 current contrast control register 1 (DCMIPP_P1CCTCR1)

Address offset: 0xAA0

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM0[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	r	r	r	r	r	r									r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM0[5:0]**: Current luminance increase for input luminance of 0 (increase is idle with LUMx = 16)

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Current ENABLE bit value

0: Bypass, pixels are forwarded idle.

1: Enable, contrast enhancement is applied on pixels.

42.14.120 DCMIPP Pipe1 current contrast control register 2 (DCMIPP_P1CCTCR2)

Address offset: 0xAA4

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM1[5:0]**: Current luminance increase for input luminance of 32 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM2[5:0]**: Current luminance increase for input luminance of 64 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM3[5:0]**: Current luminance increase for input luminance of 96 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM4[5:0]**: Current luminance increase for input luminance of 128 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

42.14.121 DCMIPP Pipe1 current contrast control register 3 (DCMIPP_P1CCTCR3)

Address offset: 0xAA8

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM5[5:0]**: Luminance increase for input luminance of 160 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM6[5:0]**: Luminance increase for input luminance of 192 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM7[5:0]**: Luminance increase for input luminance of 224 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM8[5:0]**: Luminance increase for input luminance of 256 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

42.14.122 DCMIPP Pipe1 current histogram configuration register (DCMIPP_P1CHSCR)

Address offset: 0xAB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	BIN[1:0]		DYN[1:0]		Res.	COMP[2:0]			Res.	VDEC[2:0]		
				r	r	r	r		r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HDEC[2:0]			VREG[3:0]				HREG[3:0]				SRC[2:0]			EN
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:26 **BIN[1:0]**: Bin amount per histogram. All bins have the same width.
0x0: 4 bins per histogram (assuming an input with 256 values, each bin is 64-value wide)
0x1: 16 bins per histogram (assuming an input with 256 values, each bin is 16-value wide)
0x2: 64 bins per histogram (assuming an input with 256 values, each bin is 4-value wide)
0x3: 256 bins per histogram (assuming an input with 256 values, each bin is 1-value wide)

Bits 25:24 **DYN[1:0]**: Dynamic of the pixel value, when extracting the histogram
The input pixel value is assumed mapped on 14 bits, 8 are extracted from them to select the bin to update. It is possible to map non-msb bits, to focus the histogram dynamic on the darker pixels. When non-msb bits are extracted to select a bin, if the pixel value is larger than the highest bin, then the highest bin is incremented. In other words, the highest bin collects pixel values higher than the selected dynamic.
0x0: pixel[13:6], used to extract a histogram of the light pixel values (default at reset)
0x1: pixel[11:4], used to extract a histogram of the semi light pixel values
0x2: pixel[9:2], used to extract a histogram of the semi dark pixel values
0x3: pixel[7:0], used to extract a histogram of the dark pixel values

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **COMP[2:0]**: Component selected for the histograms:
0x0: one histogram is extracted, from the R component (if pixel is RGB or raw Bayer)
0x1: one histogram is extracted, from the G (if pixel is RGB) or Gr (if raw Bayer) component
0x2: one histogram is extracted, from the B (if pixel is RGB) or Gb (if raw Bayer) component
0x3: one histogram is extracted, from the computed L (if pixel is RGB) or B (if raw Bayer)
0x4: four histograms are extracted in parallel, with R, G, B, L (if pixel is RGB), or R, Gr, Gb, B
Other values: reserved

Note: Gr means the green component among a green-red raw Bayer line, Gb means the green component among a green-blue raw Bayer line. L means the Luminance, extracted as (R/4 + G/2 + B/4).

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **VDEC[2:0]**: Vertical decimation ratio
0x0: all pixels are transmitted, no vertical decimation
0x1: one line out of 2 transmitted (if raw Bayer: 2 lines out of 4)
0x2: one line out of 4 transmitted (if raw Bayer: 2 lines out of 8)
0x3: one line out of 8 transmitted (if raw Bayer, 2 lines out of 16)
0x4: one line out of 16 transmitted (if raw Bayer, 2 lines out of 32)
Other values: reserved

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HDEC[2:0]**: Horizontal decimation ratio
0x0: all pixels are transmitted, no vertical decimation
0x1: one line out of 2 transmitted (if raw Bayer: 2 lines out of 4)
0x2: one line out of 4 transmitted (if raw Bayer: 2 lines out of 8)
0x3: one line out of 8 transmitted (if raw Bayer, 2 lines out of 16)
0x4: one line out of 16 transmitted (if raw Bayer, 2 lines out of 32)
Other values: reserved

Bits 11:8 **VREG[3:0]**: Amount of vertical regions per frame, from which to extract the histograms:
0x0: 1 region per frame
0x1: 2 regions per frame
...
0xF: 16 regions per frame

Bits 7:4 **HREG[3:0]**: Amount of horizontal regions per frame, from which to extract the histograms
0x0: 1 region per frame
0x1: 2 regions per frame
...
0xF: 16 regions per frame

Bits 3:1 **SRC[2:0]**: Source across the pipeline, where to sample pixels and extract the histograms:
0x0: after the input decimation (and before the black level)
0x1: after the black level (and before the exposure)
0x2: after the exposure (and before the demosaicing)
0x3: after the demosaicing (and before the ColorConv)
0x4: after the ColorConv (and before the contrast)
0x5: after the contrast (and before entering the applicative pipe and its CROP)
Other values: reserved
Note: When SRC ? 3, the pixel is assumed RGB.

Bit 0 **EN**: Histogram enable
0: disabled
1: enabled, histograms are extracted and written out.

42.14.123 DCMIPP Pipe1 current histogram window start register (DCMIPP_P1CHSSTR)

Address offset: 0xAB4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high.

The unit is the decimated period of pixel. Note that a decimation by 2x in raw Bayer counts two pixels every four, and VSIZE counts every four pixels.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels high.

The unit is the decimated period of pixel. Note that a decimation by 2x in raw Bayer counts two pixels every four, and HSIZE counts every four pixels.

42.14.124 DCMIPP Pipe1 current histogram window size register (DCMIPP_P1CHSSZR)

Address offset: 0xAB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size of each histogram region, from 0 to 4094 pixels high

The unit is the decimated period of pixel. Note that a decimation by 2x in raw Bayer counts two pixels every four, and VSIZE counts every four pixels.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size of each histogram region, from 0 to 4094 pixels high

The unit is the decimated period of pixel. Note that a decimation by 2x in raw Bayer counts two pixels every four, and HSIZE counts every four pixels.

42.14.125 DCMIPP Pipe1 current histogram memory address register 1 (DCMIPP_P1CHSMAR1)

Address offset: 0xABC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MA[31:0]**: Memory address

Base address of memory area, to whom histogram data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.126 DCMIPP Pipex current flow control configuration register (DCMIPP_PxCFCTCR)

Address offset: 0xB00 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame.

1: Capture requested for next frame.

When PIPEN = 1, and when the CPTREQ is set to 1 the pipe waits for the first VSync, and automatically starts a capture and sets CPTACT= 1 to mention it.

In Snapshot mode, the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode, the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMIPP and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode - Received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.
00: all frames are captured
01: one out of two frames captured (50% bandwidth reduction)
10: one out of four frames captured (75% bandwidth reduction)
11: one out of eight frames captured (87% bandwidth reduction)

42.14.127 DCMIPP Pipex current crop window start register (DCMIPP_PxCCRSTR)

Address offset: 0xB04 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 pixels wide

42.14.128 DCMIPP Pipex current crop window size register (DCMIPP_PxCCRSZR)

Address offset: 0xB08 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	VSIZE[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current ENABLE bit value.
0: Bypass, all pixels are transmitted through (no need to set any start and size values)
1: Enable, only the rectangle defined by VSIZE, HSIZE, VSTART, HSTART is transmitted.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

42.14.129 DCMIPP Pipex current decimation register (DCMIPP_PxCDCCR)

This register controls decimation, prior to downsize (to complement it).

Address offset: 0xB0C + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio
0x0: All pixels are transmitted, no vertical decimation
0x1: one line out of two transmitted
0x2: one line out of four transmitted
0x3: one line out of eight transmitted

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio
0x0: All pixels are transmitted, no horizontal decimation
0x1: one line out of two transmitted
0x2: one line out of four transmitted
0x3: one line out of eight transmitted

Bit 0 **ENABLE**:
0: Bypass: decimation is not active, all pixels are transmitted
1: Enable: one pixel every 1, 2, 4 or 8, in H and V is transmitted, following HDEC and VDEC

42.14.130 DCMIPP Pipex current downsize configuration register (DCMIPP_PxCDSCR)

Address offset: 0xB10 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The division factor is an unsigned integer based on the source/destination ratio:

div = max(int(024 / (source/destination)), 1023)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]									
r						r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current value of bit ENABLE
0: Down scaler is bypassed
1: Down scaler is enabled

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:16 **VDIV[9:0]**: Current vertical division factor, from 128 (8x) to 1023 (1x)

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **HDIV[9:0]**: Current horizontal division factor, from 128 (8x) to 1023 (1x)

42.14.131 DCMIPP Pipex current downsize ratio register (DCMIPP_PxCDSRTIOR)

Address offset: 0xB14 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The down scale ratio is expressed in unsigned 3.13 format:

ratio = max(int((source / destination) * 8192), 8 * 8192 - 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VRATIO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRATIO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **VRATIO[15:0]**: Current vertical ratio, from 8192 (1x) to 65535 (8x)

Bits 15:0 **HRATIO[15:0]**: Current horizontal ratio, from 8192 (1x) to 65535 (8x)

42.14.132 DCMIPP Pipex current downsize destination size register (DCMIPP_PxCDSSZR)

Address offset: 0xB18 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

42.14.133 DCMIPP Pipe1 current pixel packer configuration register (DCMIPP_P1CPPCR)

Address offset: 0xBC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT[3:0]				
r	r	r									r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE Event and Interrupt
0x0: Event after one line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and U-vs-V components if YUV
0: No swap of R-vs-B (U-vs-V)
1: Swap active

Bits 3:0 **FORMAT[3:0]**: Memory format
0x0: RGB888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888 (with A = 0xFF)
0x3: RGBA8888 (with A = 0xFF)
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (32 bpp, FOURCC = AYUV, with A = 0xFF)
0x6: YUV422 1-buffer (16 bpp, FOURCC = YUYV)
0x7: YUV422 2-buffer (16 bpp, FOURCC = none)
0x8: YUV420 2-buffer (12 bpp, FOURCC = NV21), NV12 available with SWAPRB = 1
0x9: YUV420 3-buffer (12 bpp, FOURCC = YV12)
0xA: YUV422 1-buffer (16 bpp, FOURCC = UYVY)
Other formats are reserved and not supported.

42.14.134 DCMIPP Pipe1 current pixel packer Memory0 address register 1 (DCMIPP_P1CPPM0AR1)

Address offset: 0xBC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of current memory area 0, to whom data are written.
It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.135 DCMIPP Pipex current pixel packer Memory0 pitch register (DCMIPP_PxCPPM0PR)

Address offset: 0xBCC + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Current number of bytes between the address of two consecutive lines.

It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

42.14.136 DCMIPP Pipex current pixel packer Memory1 address register 1 (DCMIPP_PxCPPM1AR1)

Address offset: 0xBD4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of current memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.137 DCMIPP Pipex current pixel packer Memory1 pitch register (DCMIPP_PxCPPM1PR)

Address offset: 0xBDC + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Current number of bytes between the address of two consecutive lines

It is assumed to be a multiple of 16, hence its bits [3:0] are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

42.14.138 DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR1)

Address offset: 0xBE4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of current memory area 2 where data are written. It is assumed to be a multiple of 16, hence its bits [3:0] are always at 0x0.

42.14.139 DCMIPP Pipe2 flow selection configuration register (DCMIPP_P2FSCR)

Address offset: 0xC04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
										rw	rw	rw	rw	rw	rw

Bit 31 **PIPEN**: Activation of PipeN

0: PipeN disabled

1: PipeN enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Bit 30 **FDTFEN**: Force data type format enable

0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx

1: Enable force data type format. When the data type DTIDA is detected by the pipe, the real pixel format considered by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host.

This bit has no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 29:24 **FDTF[5:0]**: Force data type format

This data type overwrites the one received by the CSI2 Host by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard.

This bit field has no meaning if the FDTFEN bit is cleared.

These bits have no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR, hence the pipe takes the value VC[1:0] configured into the DMCIPP_P1FSCR register.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type ID

Data type ID of the CSI flow to select.

These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR, hence the pipe takes the value of VC and DTIDA[5:0] / DTIDB[5:0] configured into the DMCIPP_P1FSCR register.

42.14.140 DCMIPP Pipe2 pixel packer configuration register (DCMIPP_P2PPCR)

Address offset: 0xDC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
rw	rw	rw									rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
0x0: Event after every line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and if YUV, swaps U-vs-V components
0: No swap of R-vs-B (U-vs-V)
1: Swap active.

Bits 3:0 **FORMAT[3:0]**: Memory format (only coplanar formats are supported in Pipe2)
0x0: RG/B888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888
0x3: RGBA8888
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (FOURCC = AYUV)
0x6: YUV422 1-buffer (FOURCC = YUYV)
Other formats are reserved and not supported.

42.14.141 DCMIPP Pipe2 pixel packer Memory0 address register 1 (DCMIPP_P2PPM0AR1)

Address offset: 0xDC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.142 DCMIPP Pipe2 interrupt enable register (DCMIPP_P2IER)

Address offset: 0xDF4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	Res.	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable

0: No interrupt generation

1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable

0: No interrupt generation when the line is received

1: An interrupt is generated after the full capture of a group of lines (or last line reached)

42.14.143 DCMIPP Pipe2 status register (DCMIPP_P2SR)

Address offset: 0xDF8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE
								r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
								r					r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status

0: Capture currently inactive

1: Capture currently active

This bit is automatically reset at the end of frame capture complete event (after all the data of that frame have been captured, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event.

The information is extracted from the frame data number which can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and data of this frame are corrupted

This bit is cleared by writing 1 to the COVRF bit in the DCMIPP_P2FCR register.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status

This bit is set when the VSYNC signal changes from the inactive state to the active state. In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CVSYNCF bit in the DCMIPP_P2FCR register.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P2FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status
This bit is set when one/more lines have been completed.
The periodicity of LINEF event is configured bu LINEMULT bits into DCMIPP_P1PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set.
It is cleared by writing 1 to the CLINEF bit in the DCMIPP_P2FCR register.

42.14.144 DCMIPP Pipe2 interrupt clear register (DCMIPP_P2FCR)

Address offset: 0xDFC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	Res.	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
								w					w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the OVRF bit in the DCMIPP_P2SR register.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P2SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P2SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears LINEF in the DCMIPP_P2SR register.

42.14.145 DCMIPP Pipe2 current flow selection configuration register (DCMIPP_P2CFSCR)

Address offset: 0xE04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.
r	r	r	r	r	r	r	r				r	r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
										r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN

0: Pipe disabled

1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bit 30 **FDTFEN**: Current force data type format enable

0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx

1: Enable force data type format. When data type DTIDA is detected by the pipe, the real pixel format considering by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host.

This bit has no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 29:24 **FDTF[5:0]**: Current force data type format

Forcing the data type format. This data type overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared, or if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe. These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR. For this configuration the pipe takes the value VC[1:0] configured in the DMCIPP_P1FSCR register.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type ID

Data type ID of the CSI flow to select. For this configuration the pipe takes the value DTIDA[5:0] configured in the DMCIPP_P1FSCR register.

42.14.146 DCMIPP Pipe2 current pixel packer configuration register (DCMIPP_P2CPPCR)

Address offset: 0xFC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
r	r	r									r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
0x0: Event after every line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and if YUV, swaps U-vs-V components
0: No swap of R-vs-B (U-vs-V)
1: Swap active.

Bits 3:0 **FORMAT[3:0]**: Memory format (only coplanar formats are supported in Pipe2)
0x0: RG/B888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888
0x3: RGBA8888
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (FOURCC = AYUV)
0x6: YUV422 1-buffer (FOURCC = YUYV)
Other formats are reserved and not supported.

42.14.147 DCMIPP Pipe2 current pixel packer Memory0 address register 1 (DCMIPP_P2CPPM0AR1)

Address offset: 0xFC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

42.14.148 DCMIPP hardware configuration register 3 (DCMIPP_HWCFGR3)

Address offset: 0xFE8

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LF
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LF**: Lens shading and Focus extraction
- 0: Lens shading and Focus extraction are not implemented
- 1: Lens shading and Focus extraction are implemented

42.14.149 DCMIPP hardware configuration register 2 (DCMIPP_HWCFGR2)

Address offset: 0xFEC

Reset value: 0x1103 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	STV	Res.	Res.	Res.	TPG	Res.	Res.	Res.	MCU	Res.	Res.	DEC2	DEC1
			r				r				r			r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADD MOD	Res.	Res.	Res.	PROC CLK	Res.	Res.	Res.	DBMFT	Res.	VPFT[2:0]		
			r				r				r		r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **STV**: Statistic version
- 0: Statistic extraction (with three accumulators) is implemented
- 1: Statistic and histogram extractions are implemented

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TPG**: Test pattern generator
- 0: TPG is not implemented
- 1: TPG is implemented

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **MCU**: Macroblock unit as pixel format
- 0: MCU is not implemented
- 1: MCU is implemented

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **DEC2**: Decimation on Pipe2 before downsize
- 0: Decimation is not implemented
- 1: Decimation is implemented

Bit 16 **DEC1**: Decimation on Pipe1 before downsize
0: Decimation is not implemented
1: Decimation is implemented

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ADDMOD**: Address modulo computation to access a small buffer in streaming featured
0: Address modulo is not implemented
1: Address modulo is implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **PROCCLK**: Processing clock linked to AXI clock featured
0: Independent processing clock not connected to AXI
1: Processing clock is directly the AXI

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **DBMFT**: Double buffer mode featured

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **VPFT[2:0]**: Virtual pipe function

42.14.150 DCMIPP hardware configuration register 1 (DCMIPP_HWCFGR1)

Address offset: 0xFF0

Reset value: 0x0031 3131

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROI2NB[3:0]				ROI1NB[3:0]				Res.	Res.	PLANARFT[1:0]		Res.	Res.	Res.	RB2 RGB
r	r	r	r	r	r	r	r			r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DSP2 FT	DSP1 FT	Res.	Res.	Res.	IPPLUG CFG	Res.	Res.	PIPENB[1:0]		Res.	Res.	Res.	CSIFT
		r	r				r			r	r				r

Bits 31:28 **ROI2NB[3:0]**: Number of ROIs for Pipe2

Bits 27:24 **ROI1NB[3:0]**: Number of ROIs for Pipe1

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **PLANARFT[1:0]**: Buffer features for Pipe1
00: Coplanar
01: Semiplanar
10: Reserved
11: Full planar (one buffer per component)

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RB2RGB**: Raw Bayer to RGB feature (demosaicer)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DSP2FT**: Down-sampling feature for the pixel Pipe2

Bit 12 **DSP1FT**: Down-sampling feature for the pixel Pipe1

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **IPPLUGCFG**: IP-Plug configuration

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PIPENB[1:0]**: Number of pipes

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CSIFT**: CSI2 host protocol compliant

42.14.151 DCMIPP version register (DCMIPP_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: DCMIPP major revision

Bits 3:0 **MINREV[3:0]**: DCMIPP minor revision

42.14.152 DCMIPP identification register (DCMIPP_IPIDR)

Address offset: 0xFF8

Reset value: 0x0016 2021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IDR[31:0]**: Parallel camera interface (DCMI) and optional pixel processing (PP)

42.14.153 DCMIPP size identification register (DCMIPP_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: 4-Kbyte decoding space

42.15 DCMIPP register map

Table 413. DCMIPP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DCMIPP_IPGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QOS_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMORYPAGE [2:0]		
	Reset value								0																						0	1	0
0x004	**DCMIPP_IPGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTART
	Reset value																																0
0x008	**DCMIPP_IPGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLE
	Reset value																																1
0x00C-0x018	Reserved	Reserved																															
0x01C	**DCMIPP_IPGR8**	IPPID[7:0]								Res.	Res.	Res.	ARCHIID[4:0]					Res.	Res.	Res.	REVID[4:0]					Res.	Res.	DID[5:0]					
	Reset value	1	0	1	0	1	0	1	0				0	0	1	0	0				0	0	0	1	1			0	1	0	1	0	0
0x020	**DCMIPP_IPC1R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC [2:0]		
	Reset value																					0	0	0	0						1	0	0
0x028	**DCMIPP_IPC1R3**	Res.	Res.	Res.	Res.	Res.	DPREGEND[10:0]											Res.	Res.	Res.	Res.	DPREGSTART[10:0]											
	Reset value						0	0	0	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	**DCMIPP_IPC2R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR [2:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC [2:0]		
	Reset value																					0	0	0							1	0	0
0x034	**DCMIPP_IPC2R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]				Res.	Res.	Res.	Res.	SVC MAPPING [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x038	**DCMIPP_IPC2R3**	Res.	Res.	Res.	Res.	Res.	DPREGEND[10:0]											Res.	Res.	Res.	Res.	DPREGSTART[10:0]											
	Reset value						0	0	0	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	**DCMIPP_IPC3R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR [2:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC [2:0]		
	Reset value																					0	0	0							1	0	0
0x044	**DCMIPP_IPC3R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]				Res.	Res.	Res.	Res.	SVC MAPPING [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x048	**DCMIPP_IPC3R3**	Res.	Res.	Res.	Res.	Res.	DPREGEND[10:0]											Res.	Res.	Res.	Res.	DPREGSTART[10:0]											
	Reset value						0	0	0	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	**DCMIPP_IPC4R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR [2:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC [2:0]		
	Reset value																					0	0	0							1	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	**DCMIPP_IPC4R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]				Res.	Res.	Res.	Res.	SVC MAPPING [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x058	**DCMIPP_IPC4R3**	Res.	Res.	Res.	Res.	Res.	DPREGEND[10:0]											Res.	Res.	Res.	Res.	Res.	DPREGSTART[10:0]										
	Reset value							0	0	0	1	1	1	1	1	1	1						0	0	0	0	0	0	0	0	0	0	0
0x05C to 0x100	Reserved	Reserved																															
0x100	**DCMIPP _PRHWCFGR**	PRHWCFGR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**DCMIPP_PRCR**	Res.	Res.	Res.	Res.	Res.	SWAPBITS	SWAPCYCLES	Res.	FORMAT[7:0]								Res.	ENABLE	Res.	EDM [2:0]			Res.	Res.	VSPOL	HSPOL	PCKPOL	ESS	Res.	Res.	Res.	Res.
	Reset value						0	0		0	0	0	0	0	0	0	0		0		0	0	0			0	0	0	0				
0x108	**DCMIPP_PRESCR**	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	**DCMIPP_PRESUR**	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110 to 0x1F0	Reserved	Reserved																															
0x1F4	**DCMIPP_PRIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0						
0x1F8	**DCMIPP_PRSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSYNC	HSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															1	1										0						
0x1FC	**DCMIPP_PRFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0						
0x200	**DCMIPP_ CMHWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPG	CCSI	CPAR	CPIPES [1:0]		Res.	Res.
	Reset value																										0	1	1	1	1		
0x204	**DCMIPP_CMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	Res.	Res.	CFC	Res.	PSFC [1:0]		INSEL
	Reset value																									0			0		0	0	0
0x208	**DCMIPP_CMFRCR**	FRMCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20C	Reserved																																
0x210	**DCMIPP_ CMTPGCR1**	Res.	Res.	HEIGHT[13:0]														Res.	Res.	WIDTH[13:0]														
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x214	**DCMIPP_ CMTPGCR2**	VBL[15:0]																FORMAT[7:0]								RT [1:0]		YT [1:0]		PATTERN	GSEN	LFLEN	TPGEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218-0x3EC	Reserved	Reserved																															

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F0	**DCMIPP_CMIER**		Res.	Res.	Res.	Res.					STOVRIE	STBCRIE	STFRAMEIE	Res.				P0OVRIE	P0LIMITIE	Res.	Res.	Res.	P0VSYNCIE	P0FRAMEIE	P0LINEIE	Res.	PRERRIE	ATXERRIE	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0					0	0				0	0	0		0	0					
0x3F4	**DCMIPP_CMSR1**	P2CPTACT	Res.	Res.	Res.	Res.	Res.	P2LSTFRM	P2LSTLINE	P1CPTACT	Res.	Res.	Res.	Res.	Res.	P1LSTFRM	P1LSTLINE	P0CPTACT	Res.	Res.	Res.	Res.	Res.	P0LSTFRM	P0LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	PRVSYNC	PRHSYNC
	Reset value	0						0	0	0						0	0	0						0	0							1	1
0x3F8	**DCMIPP_CMSR2**		Res.	Res.	Res.	Res.					STOVRF	STBCRF	STFRAMEF	Res.				P0OVRF	P0LIMITF	Res.	Res.	Res.	P0VSYNCF	P0FRAMEF	P0LINEF	Res.	PRERRF	ATXERRF	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0					0	0				0	0	0		0	0					
0x3FC	**DCMIPP_CMFCR**		Res.	Res.	Res.	Res.					STOVRF	STBCRF	STFRAMEF	Res.				CP0OVRF	CP0LIMITF	Res.	Res.	Res.	CP0VSYNCF	CP0FRAMEF	CP0LINEF	Res.	CPRERRF	CATXERRF	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0					0	0				0	0	0		0	0					
0x400	**DCMIPP_ P0HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CROI [3:0]				CGM[1:0]		CRB[1:0]		DBM	CDS	CPLA[1:0]	
	Reset value																					0	0	0	0	0	0	0	0	1	0	0	0
0x404	**DCMIPP_P0FSCR**	PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC [1:0]		Res.	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
	Reset value	0											0	0		0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x408-0x4FC	Reserved																																
0x500	**DCMIPP_P0FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x504	**DCMIPP_P0SCSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x508	**DCMIPP_P0SCSZR**	ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x50C-0x5AC	Reserved	Reserved																															
0x5B0	**DCMIPP_P0DCCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	CNT[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B4	**DCMIPP_P0DCLMTR**	ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIMIT[23:0]																							
	Reset value	0								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5B8 - 0x5BC	Reserved	Reserved																															

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C0	**DCMIPP_P0PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADEREN	PAD	Res.	Res.	Res.	Res.	SWAPYUV
	Reset value																	0	0	0		0	0	0	0	0	0	0					0
0x5C4	**DCMIPP _P0PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	**DCMIPP _P0STM0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8-0x5F0	Reserved	Reserved																															
0x5F4	**DCMIPP_P0IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	LIMITIE	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0	0				0	0	0
0x5F8	**DCMIPP_P0SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	Res.	LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	LIMITF	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0							0									0	0				0	0	0
0x5FC	**DCMIPP_P0FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CLIMITF	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0	0				0	0	0
0x600	Reserved	Reserved																															
0x604	**DCMIPP_P0CFSCR**	PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC [1:0]		Res.	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]						
	Reset value	0											0	0		0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x608-0x6FC	Reserved	Reserved																															
0x700	**DCMIPP_P0CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x704	**DCMIPP_P0CSCSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x708	**DCMIPP_P0CSCSZR**	ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x70C-0x7BC	Reserved	Reserved																															
0x7C0	**DCMIPP_P0CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADEREN	PAD	Res.	Res.	Res.	Res.	SWAPYUV
	Reset value																	0	0	0		0	0	0	0	0	0	0					0
0x7C4	**DCMIPP _P0CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C8	DCMIPP _P0CPPM0AR2	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7CC - 0x7FC	Reserved	Reserved																															
0x800	**DCMIPP_ P1HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFOC	CLS	CHS	Res.	Res.	Res.	CDEC	CROI [3:0]				CGM[1:0]		CRB[1:0]		DBM	CDS	CPLA[1:0]	
	Reset value														0	0	0				0	0	0	0	0	0	1	0	1	0	1	1	0
0x804	**DCMIPP_P1FSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC [1:0]		PIPEDIFF	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x810- 0x81C	Reserved	Reserved																															
0x820	**DCMIPP_P1SRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CROPEN	FIRSTLINE DEL[2:0]		LASTLINE[11:0]												
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x824	**DCMIPP_P1BPRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGHT [2:0]			ENABLE
	Reset value																													0	0	0	0
0x828	**DCMIPP_P1BPRSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BADCNT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x82C	Reserved	Reserved																															
0x830	**DCMIPP_P1DECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
	Reset value																												0	0	0	0	0
0x834- 0x83C	Reserved	Reserved																															
0x840	**DCMIPP_P1BLCCR**	BLCR[7:0]								BLCG[7:0]								BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0x844	**DCMIPP_P1EXCR1**	Res.	SHFR [2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value		0	0	0	0	0	0	0	0	0	0	0																				0
0x848	**DCMIPP_P1EXCR2**	Res.	SHFG [2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.	Res.	SHFB [2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0				
0x84C	Reserved	Reserved																															
0x850	**DCMIPP_P1ST1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0
0x854	**DCMIPP_P1ST2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0
0x858	**DCMIPP_P1ST3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x85C	**DCMIPP_P1STSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x860	**DCMIPP_P1STSZR**	CROPEN	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x864	DCMIPP_P1ST1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x868	**DCMIPP_P1ST2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x86C	**DCMIPP_P1ST3SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x870	**DCMIPP_P1DMCR**	Res.	EDGE [2:0]			Res.	LINEH [2:0]			Res.	LINEV [2:0]			Res.	PEAK [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE [1:0]		ENABLE
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0														0	0	0
0x874-0x87C	Reserved	Reserved																															
0x880	**DCMIPP_P1CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0x884	**DCMIPP_P1CCRR1**	Res.	Res.	Res.	Res.	Res.	RG[10:0]											Res.	Res.	Res.	Res.	Res.	RR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x888	**DCMIPP_P1CCRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	Res.	RB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x88C	**DCMIPP_P1CCGR1**	Res.	Res.	Res.	Res.	Res.	GG[10:0]											Res.	Res.	Res.	Res.	Res.	GR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x890	**DCMIPP_P1CCGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x894	**DCMIPP_P1CCBR1**	Res.	Res.	Res.	Res.	Res.	BG[10:0]											Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x898	**DCMIPP_P1CCBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x89C	Reserved	Reserved																															
0x8A0	**DCMIPP_P1CTCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUM0[5:0]														ENABLE
	Reset value																		0	1	0	0	0	0									0
0x8A4	**DCMIPP_P1CTCR2**	Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]							Res.	Res.	LUM3[5:0]					Res.	Res.	LUM4[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0				0	1	0	0	0	0			0	1	0	0	0	0	
0x8A8	**DCMIPP_P1CTCR3**	Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]							Res.	Res.	LUM7[5:0]					Res.	Res.	LUM8[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0				0	1	0	0	0	0			0	1	0	0	0	0	
0x8AC	Reserved	Reserved																															
0x8B0	**DCMIPP_P1HSCR**	Res.	Res.	Res.	Res.	BIN [1:0]		BIN [1:0]		Res.	COMP [2:0]			Res.	VDEC [2:0]			Res.	HDEC [2:0]			VREG [3:0]				VREG [3:0]				SRC [2:0]			EN
	Reset value					0	0	0	0		0	0	0		0	0	1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8B4	**DCMIPP_P1HSSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x8B8	**DCMIPP_P1HSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x8BC	**DCMIPP_P1HSMAR1**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C4	**DCMIPP_P1HSSMAR**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8D0-0x8FC	Reserved	Reserved																															
0x900	**DCMIPP_P1FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x904	**DCMIPP_P1CRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x908	**DCMIPP_P1CRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x90C	**DCMIPP_P1DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]	ENABLE
																														0	0	0	0	0
0x910	**DCMIPP_P1DSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x914	**DCMIPP_P1DSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	**DCMIPP_P1DSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x91C-0x96C	Reserved	Reserved																															
0x970	**DCMIPP_P1GMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0x974-0x97C	Reserved	Reserved																															
0x980	**DCMIPP_P1YUVCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0x984	**DCMIPP_P1YUVRR1**	Res.	Res.	Res.	Res.	Res.	RG[10:0]											Res.	Res.	Res.	Res.	RR[10:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x988	**DCMIPP_P1YUVRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	RB[10:0]											
	Reset value							0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x98C	**DCMIPP_P1YUVGR1**	Res.	Res.	Res.	Res.	Res.	GG[10:0]											Res.	Res.	Res.	Res.	GR[10:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x990	**DCMIPP_P1YUVGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x994	**DCMIPP_P1YUVBR1**	Res.	Res.	Res.	Res.	Res.	BG[10:0]											Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x998	**DCMIPP_P1YUVBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x99C-0x9BC	Reserved	Reserved																															
0x9C0	**DCMIPP_P1PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT [3:0]			
	Reset value																	0	0	0									0	0	0	0	0
0x9C4	**DCMIPP_P1PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C8	Reserved	Reserved																															
0x9CC	**DCMIPP_P1PPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D0	**DCMIPP_P1STM0A**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D0	Reserved	Reserved																															
0x9D4	**DCMIPP_P1PPM1AR1**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D8	Reserved	Reserved																															
0x9DC	**DCMIPP_P1PPM1PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E0	**DCMIPP_P1STM1AR**	M1A[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E0	Reserved	Reserved																															
0x9E4	**DCMIPP_P1PPM2AR1**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9F0	**DCMIPP_P1STM2AR**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E8-0x9F0	Reserved	Reserved																															
0x9F4	**DCMIPP_P1IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	STOVRIE	STBCIE	STFRAMEIE	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0	0	0	0		0	0	0
0x9F8	**DCMIPP_P1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	STOVRF	STBCF	STFRAMEF	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0																0	0	0	0		0	0	0
0x9FC	**DCMIPP_P1FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CSTOVRF	CSTBCF	CSTFRAMEF	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0	0	0	0		0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA00	Reserved	Reserved																															
0xA04	**DCMIPP_P1CFSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC [1:0]		PIPEDIFF	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0xA08-0xA20	Reserved	Reserved																															
0xA24	**DCMIPP_P1CBPRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH [2:0]			ENABLE
	Reset value																													0	0	0	0
0xA28-0xA3C	Reserved	Reserved																															
0xA40	**DCMIPP_P1CBLCCR**	BLCR[7:0]								BLCG[7:0]								BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0xA44	**DCMIPP_P1CEXCR1**	Res.	SHFR [2:0]			MULTR[7:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value		0	0	0	0	0	0	0	0	0	0	0																				0
0xA48	**DCMIPP_P1CEXCR2**	Res.	SHFG [2:0]			MULTG[7:0]									Res.	Res.	Res.	Res.	Res.	SHFB [2:0]			MULTB[7:0]							Res.	Res.	Res.	Res.	
	Reset value		0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0				
0xA4C	Reserved	Reserved																															
0xA50	**DCMIPP_P1CST1CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA54	**DCMIPP_P1CST2CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA58	**DCMIPP_P1CST3CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA5C	**DCMIPP_P1CSTSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA60	**DCMIPP_P1CSTSZR**	CROPEN	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA64-0xA7C	Reserved	Reserved																															
0xA80	**DCMIPP_P1CCCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0xA84	**DCMIPP_P1CCCRR1**	Res.	Res.	Res.	Res.	Res.	RG[10:0]											Res.	Res.	Res.	Res.	Res.	RR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA88	**DCMIPP_P1CCCRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	Res.	RB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA8C	**DCMIPP_P1CCCGR1**	Res.	Res.	Res.	Res.	Res.	Res.	GG[10:0]										Res.	Res.	Res.	Res.	Res.	GR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA90	**DCMIPP_P1CCCGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA94	**DCMIPP_P1CCCBR1**	Res.	Res.	Res.	Res.	Res.	Res.	BG[10:0]										Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA98	**DCMIPP_P1CCCBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA9C	Reserved	Reserved																															
0xAA0	**DCMIPP_P1CCTCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUM0[5:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																		0	1	0	0	0	0									0
0xAA4	**DCMIPP_P1CCTCR2**	Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]							Res.	Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0xAA8	**DCMIPP_P1CCTCR3**	Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]							Res.	Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0xAAC	Reserved	Reserved																															
0xAB0	**DCMIPP_P1CHSCR**	Res.	Res.	Res.	Res.	BIN [1:0]		BIN [1:0]		Res.	COMP [2:0]			Res.	VDEC [2:0]			Res.	HDEC [2:0]			VREG [3:0]				VREG [3:0]				SRC [2:0]			EN
	Reset value					0	0	0	0		0	0	0		0	0	1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAB4	**DCMIPP_P1CHSSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xAB8	**DCMIPP_P1CHSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xABC	**DCMIPP _P1CHSMAR1**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAC0	**DCMIPP _P1CHSMAR2**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAC4-0xAFC	Reserved	Reserved																															
0xB00	**DCMIPP_P1CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0xB04	**DCMIPP_P1CCRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB08	**DCMIPP_P1CCRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB0C	**DCMIPP_P1CDCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																													0	0	0	0	0
0xB10	**DCMIPP_P1CDSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0xB14	**DCMIPP _P1CDSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB18	**DCMIPP_P1CDSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB1CxBBC	Reserved	Reserved																															
0xBC0	**DCMIPP_P1CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT [3:0]			
	Reset value																	0	0	0									0	0	0	0	0
0xBC4	**DCMIPP _P1CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC8	Reserved	Reserved																															
0xBCC	**DCMIPP _P1CPPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBD0	Reserved																																
0xBD4	**DCMIPP _P1CPPM1AR1**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBD8	Reserved	Reserved																															
0xBDC	**DCMIPP _P1CPPM1PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBE0	Reserved																																
0xBE4	**DCMIPP _P1CPPM2AR1**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBE8 - 0xBFC	Reserved	Reserved																															
0xC00	**DCMIPP _P2HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDEC	CROI[3:0]				CGM [1:0]		CRB[1 :0]		DBM	CDS	CPLA [1:0]
	Reset value																				1	0	0	0	0	0	1	0	0	0	1	0	0
0xC04	**DCMIPP_P2FSCR**	PIPEN	FDTFEN	FDTF[5:0]					Res.	Res.	Res.	VC [1:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]				
	Reset value	0	0	0	0	0	0	0	0				0	0															0	0	0	0	0	0
0xC08 - 0xCFC	Reserved	Reserved																															
0xD00	**DCMIPP_P2FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD04	**DCMIPP_P2CRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD08	**DCMIPP_P2CRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD0C	**DCMIPP_P2DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																													0	0	0	0	0
0xD10	**DCMIPP_P2DSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0xD14	**DCMIPP _P2DSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD18	**DCMIPP_P2DSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD1C-0xD6C	Reserved	Reserved																															
0xD70	**DCMIPP_P2GMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0xD74-0xDBC	Reserved	Reserved																															
0xDC0	**DCMIPP_P2PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT [3:0]			
	Reset value																	0	0	0									0	0	0	0	0
0xDC4	**DCMIPP _P2PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDC8	Reserved	Reserved																															
0xDCC	**DCMIPP_P2PPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDD0-0xDF0	Reserved	Reserved																															
0xDF4	**DCMIPP_P2IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	Res.	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0					0	0	0
0xDF8	**DCMIPP_P2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0						0	0									0					0	0	0
0xDFC	**DCMIPP_P2FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	Res.	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0					0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE00	Reserved	Reserved																															
0xE04	**DCMIPP_P2CFSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0														0	0	0	0	0	0
0xE08-0xEFC	Reserved	Reserved																															
0xF00	**DCMIPP_P2CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0xF04	**DCMIPP_P2CCRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF08	**DCMIPP_P2CCRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF0C	**DCMIPP_P2DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																												0	0	0	0	0	
0xF10	**DCMIPP_P2CDSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0xF14	**DCMIPP_P2CDSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF18	**DCMIPP_P2CDSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF1C-0xFBC	Reserved	Reserved																															
0xFC0	**DCMIPP_P2CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINEMULT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT [3:0]			
	Reset value																	0	0	0									0	0	0	0	0
0xFC4	**DCMIPP_P2CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFC8	Reserved																																
0xFCC	**DCMIPP_P2CPPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFD0-0xFE4	Reserved	Reserved																															
0xFE8	**DCMIPP_HWCFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LF
	Reset value																																
0xFEC	**DCMIPP_HWCFGR2**	Res.	Res.	Res.	STV	Res.	Res.	Res.	TPG	Res.	Res.	Res.	MCU	Res.	Res.	DEC2	DEC1	Res.	Res.	Res.	ADDMOD	Res.	Res.	Res.	PROCCLK	Res.	Res.	Res.	DBMFT	Res.	VPFT [2:0]		
	Reset value				1				1				1			1	1				1				0				0		0	0	0

Table 413. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	**DCMIPP_HWCFGR1**	ROI2NB [3:0]				ROI1NB [3:0]				Res.	Res.	PLANARTF [1:0]		Res.	Res.	Res.	RB2RGB	Res.	Res.	DSP2FT	DSP1FT	Res.	Res.	Res.	IPPLUGCFG	Res.	Res.	PIPENB [1:0]		Res.	Res.	Res.	CSIFT
	Reset value		0	0	0		0	0	0			1	1				1			1	1				0			1	1				1
0xFF4	**DCMIPP_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	1	0	0	0	1
0xFF8	**DCMIPP_IPIDR**	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1
0xFFC	**DCMIPP_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

43 CSI-2 Host (CSI)

43.1 CSI introduction

The camera serial interface 2 (CSI-2) is a part of a group of communication protocols defined by the MIPI® Alliance. The MIPI® CSI-2 Host controller is a digital core that implements all protocol functions defined in the MIPI® CSI-2 specification.

It provides an interface between the system and the MIPI® D-PHY, allowing communication with a CSI-2 compliant camera.

43.2 Standard and references

- MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2) v1.3 - 29 May 2014
- MIPI Alliance Specification for D-PHY v1.2 - 01 August 2014

43.3 Glossary

Table 414. Glossary of terms

Term	Definition
D-PHY_RX	D-PHY physical layer receiver
EOF	End of frame
EOT	End of transmission
ESC	Escape mode for the D-PHY_RX
HS	High speed
ISB	Imaging system bus
LLP	Low-level protocol
LP	Low power
PHY	Physical layer
PPI	Phy protocol interface
RX	Receiver
SOF	Start of frame
SOT	Start of transmission

43.4 CSI-2 Host main features

- Compliant with MIPI Alliance standard (see *Section 43.2*)
- Interface with MIPI D-PHY (see *Section 43.2*)
- Up to two D-PHY data lanes supported (up to 2.5 Gbit/s per lane in high-speed (HS) mode and 10 Mbit/s in low-power (LP) mode)
- Data transmission in HS and LP modes supported by the D-PHY
- Escape mode (ESC) and ultra-low-power state mode (ULPS) supported
- CSI-2 lane management layer connected to the D-PHY_RX to merge 1 or 2 data lanes in a single 32-bit ISB-Byte bus
- ECC and error correction capabilities
- CRC check capability
- CSI-2 virtual channel and data type filtering supporting interleaved data
  - Up to 4 virtual channels
  - Support the data formats specified into the MIPI Alliance standard for CSI-2 v1.3 (18 data formats, plus the user defined ones)
- Up to 7 independent data types
- Programmable interrupts:
  - 4 timer interrupts mapped on a selected virtual channel with starting point from SOF (start of frame) or EOF (end of frame)
  - 4 line/byte counter interrupts mapped on a selected virtual channel to trigger an event
- ISB-Byte header generation for internal connection with the DCMIPP peripheral

43.5 CSI-2 Host functional description

43.5.1 General description

The MIPI CSI-2 Host controller offers a means to connect cameras to capture and dispatch a robust header with ECC correction and data payload with checksum.
A D-PHY_RX receiver fills out the role of interfacing the external serial camera devices with the embedded CSI-2 Host controller:

- D-PHY receiver:

  This is a D-PHY_RX receiver composed of two unidirectional data lanes, and one clock lane. It operates in slave mode with HS and LP mode capabilities. The D-PHY drives a PPI interface to communicate with the CSI-2 Host.
- CSI-2 lane merger:

  Up to two data lane inputs to increase data rate while avoiding high clock rates

  The lane manager function collects incoming bytes from N (N = 1 to 2) lanes.
  It consolidates them by merging operations into a complete packet provided to the CSI-2 low-level protocol decoder (LLP), to decompose the meaning of each data packet thanks to the CSI-2 protocol specification.
- CSI-2 clock changer:

  The clock changer moves the data transfer from an input clock domain from D-PHY_RX to a clock_proc domain before being processed by the LLP.

- CSI-2 low-level protocol decoder (LLP)

  The low-level protocol is a byte-orientated packet-based protocol that supports the transport of arbitrary data using short and long packet formats.

  It generates an ISB-byte header data for further processing, with data integrity check or miscellaneous integrity checks.

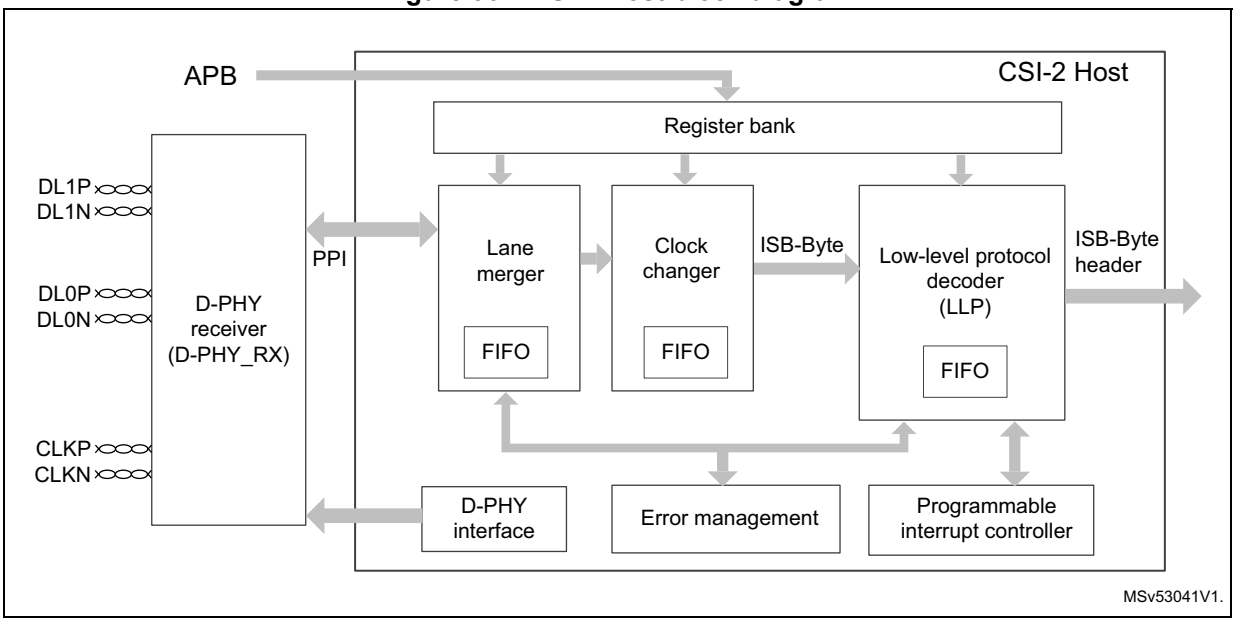
  It supports data interleaving through the support of up to four virtual channels and many data type formats.

  The LLP offers interrupt generation capability, depending on specific events such as line/byte detection within a frame, or a time-base event selected from a SOF or an EOF.

The CSI-2 Host-controller register bank is configured and accessed by the APB bus interface.

A block diagram of the CSI-2 Host is shown in the figure below.

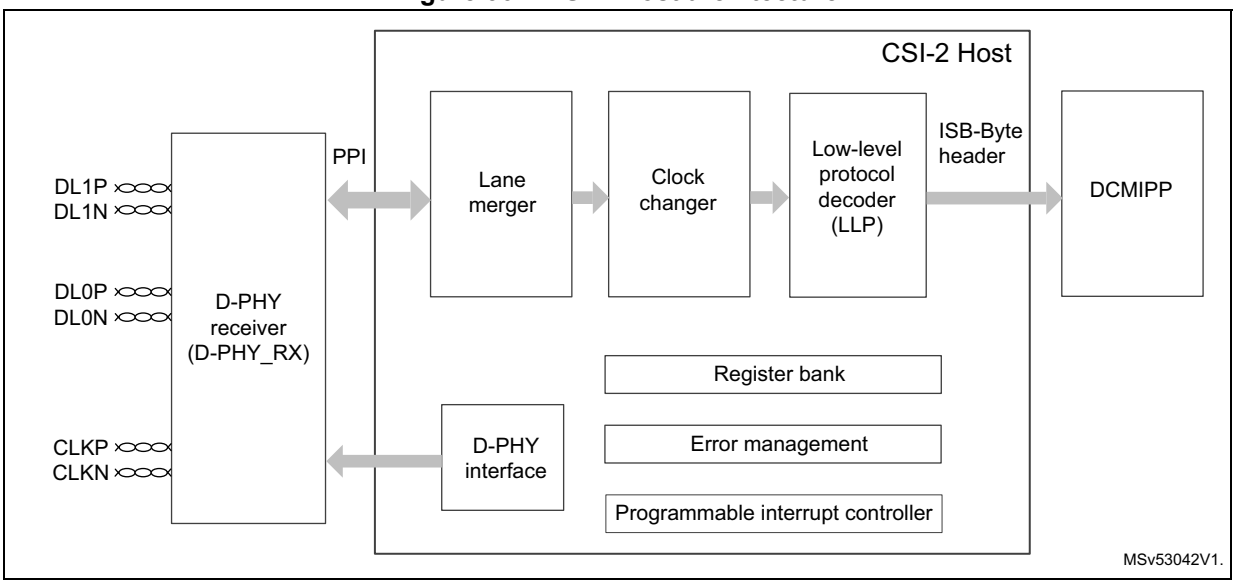
Figure 361. CSI-2 Host block diagram



43.5.2 System level architecture

The figure below shows the CSI-2 Host architecture.

Figure 362. CSI-2 Host architecture



The blocks shown have the following functions:

- D-PHY_RX is a slave receiver device that connects external serial cameras to the CSI-2 Host controller. It is based on unidirectional slave data lanes and clock lanes to handle data transmission from the CSI-2 protocol-based camera devices to subsystems for data processing (handled by the DCMIPP).
- The D-PHY interface manages D-PHY_RX register accesses. It also allows the block to be configured to raise interrupts on the detection of some errors during reception. It highlights the status of the physical lines up to the software layers.
- The lane merger module packs the data coming from the two D-PHY_RX data lanes, if required to do so for increased data rates. It generates a data flow based on an ISB-Byte protocol, which is not linked to any protocol at this point. The data is only identified as a control or a data packet for the clock changer module.
- The clock changer module manages data crossing from the D-PHY_RX clock domain to the clk_proc clock domain.
- The LLP handles data filtering to serve the data-interleaving capability of the DCMIPP. It can filter data from external devices depending on virtual channel identifiers, or on data-type format selection. It also executes, depending on the input pixel format, data reordering in order to optimize the data storage, for instance in memories.
- The CSI-2 Host manages also various errors coming (not exclusively) from the D-PHY_RX physical layer. It can detect error conditions in the received header or a short packet in a frame. ECC is implemented to correct one bit error in the header. A CRC checksum is also implemented to detect any data corruption during the data payload reception. The 16-bit CRC value is located into the packet footer.
  The CSI-2 Host also ensures that specific timings within the data reception are properly in line with the CSI-2 specification. All error conditions and associated interrupts are detailed in *Section 43.8*.

- A specific module generates interrupts following the detection of selected events in a frame. For a given virtual channel, these events may be triggers based on line- and byte-number detection within the frame, or counters from dedicated timers reaching a number of cycles (clk_proc) from an SOF or an EOF.
- Data is filtered out by the LLP, and sent to the DCMIPP in a format that is quite similar to the CSI-2 protocol, but without CRC (packet footer) and ECC information inside the packet header. The data type and the virtual channel are also sent to the DCMIPP to handle the interleaved transmission of different image data formats within the same video data stream.

43.5.3 CSI-2 Host reset and clocks

There are two reset sources:

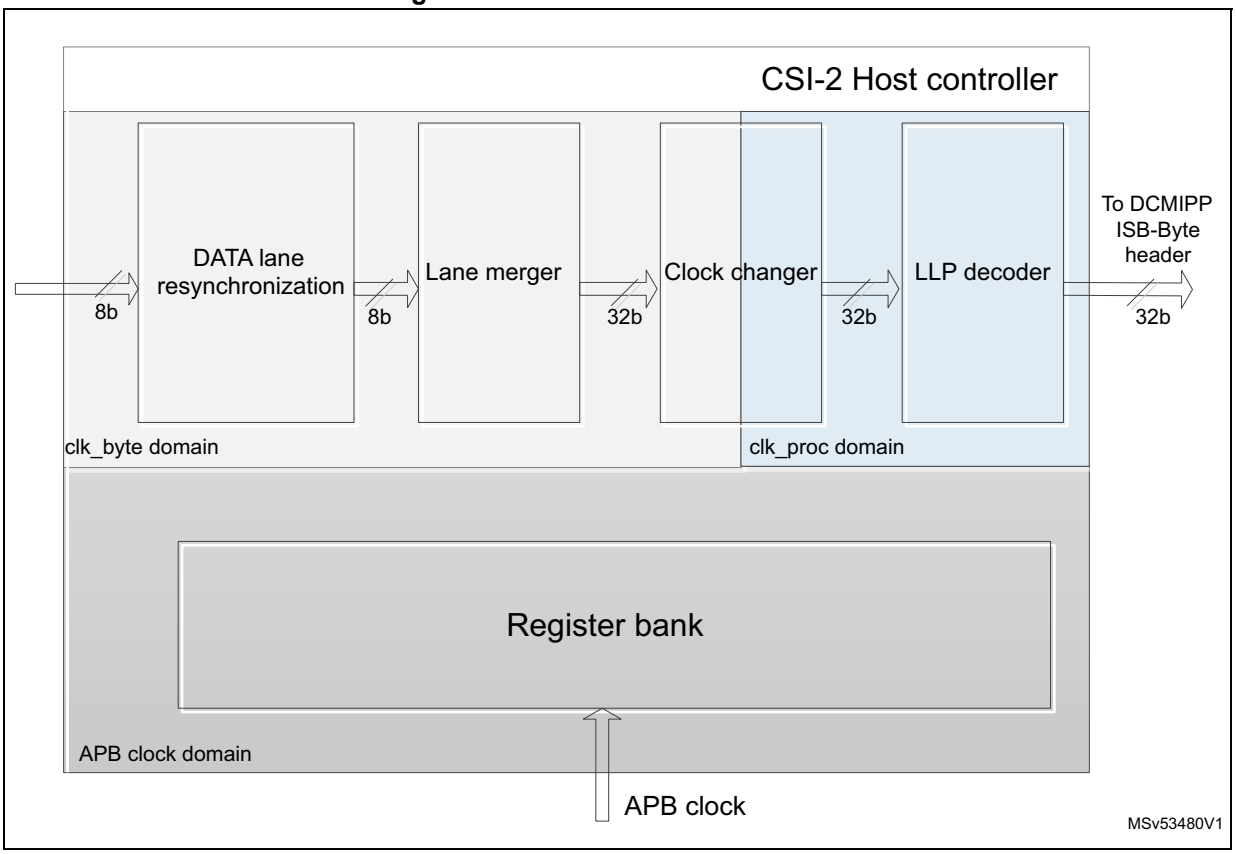
- hardware asynchronous reset which is resynchronized on clk_byte and clk_proc clock domain (processing clock) to reset the flops connected to these clock domains
- APB reset from the reset and clock controller unit to reset mainly all the registers

There are three clocks to serve the CSI-2 Host controller (see *Figure 363*):

- clk_byte: derived from the D-PHY_RX clock lane, and used by the lane merger module and the clock changer module
- clk_proc: used by the kernel of the LLP

  Resynchronization between the clk_byte domain and the clk_proc domain is handled by the clock changer module.
- PCLK clock: APB clock used to access to the register bank

Figure 363. CSI-2 Host clock domain



The D-PHY_RX supports up to 2.5 Gbit/s per data lane: the maximum frequency on the D-PHY_RX clock is 2.5 Gbits / 8 to output a byte = 312 MHz.

43.5.4 Lane merger

Applications requiring more bandwidth than the one provided by a single data lane may expand the data path to more lanes in order to increase the attainable data rate. The lane merger supports then up to two data lanes.

The lane merger module converts the D-PHY_RX PPI protocol into a byte-oriented bus to be sent to the LLP.

The lane merger parallelizes the bytes to 32 bits in parallel to build a 32-bit output as shown in the figures below. The cycle number does not account for module internal latency.

Figure 364. Lane merger input and output (one data lane)

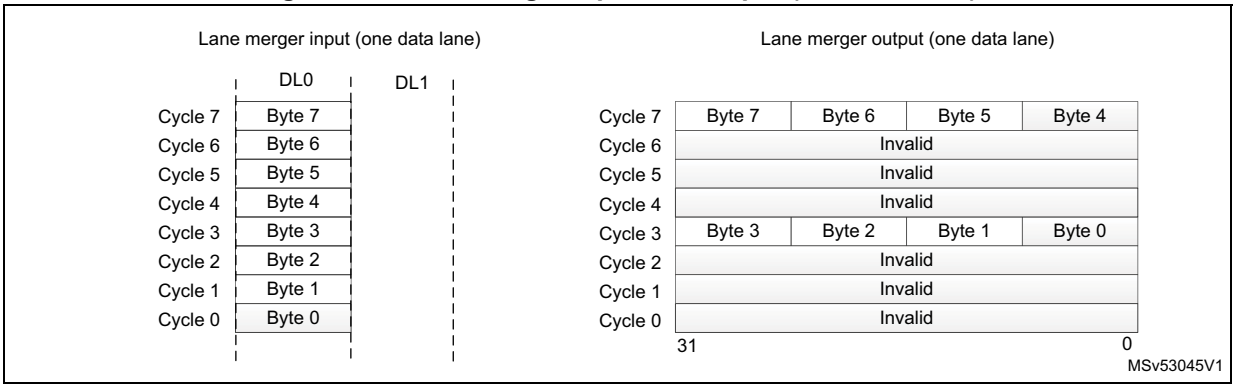
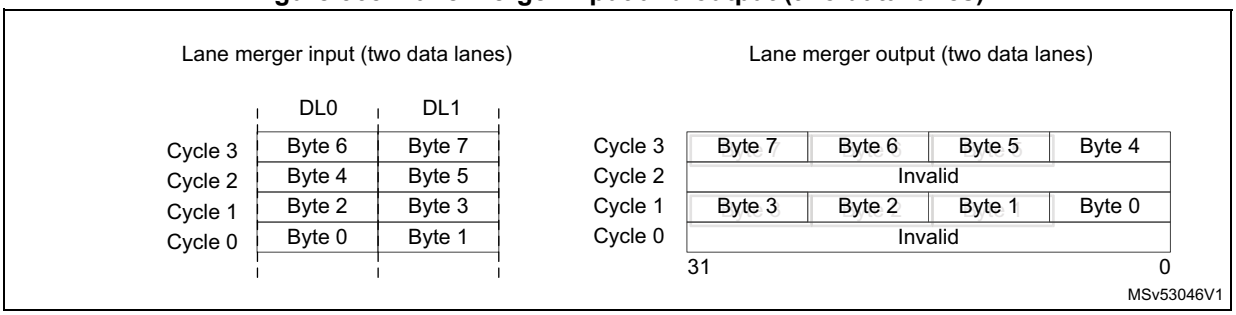
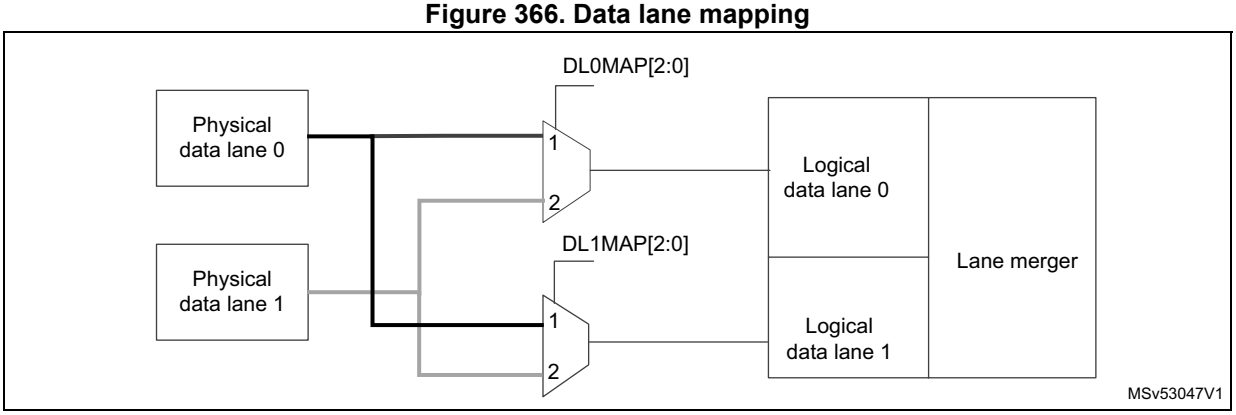


Figure 365. Lane merger input and output (two data lanes)



LANENB[2:0] in CSI_LMCFGR configures the number of data lanes used in the application. The software needs to map the physical data lanes input to the lane merger logical data lanes, using DL0MAP[2:0] and DL1MAP[2:0] as shown in the figure below.

Figure 366. Data lane mapping



43.5.5 Clock changer

The clock changer module changes the clock frequency from the input clock domain linked to the D-PHY_RX PPI protocol, to the output clock domain based on clk_proc clock domain (reference clock for the LLP).

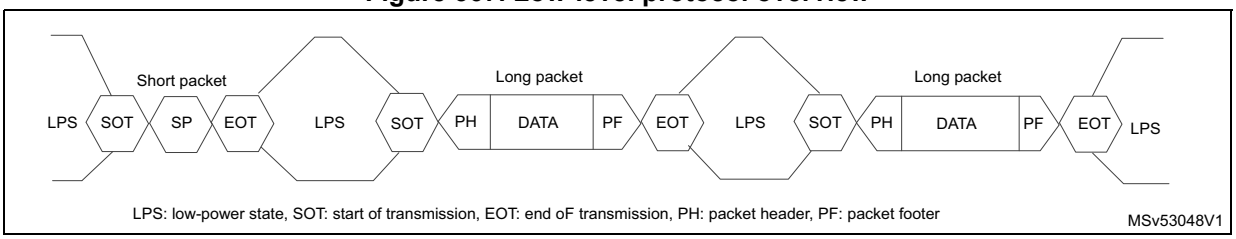
An integrated FIFO receives data from the input clock domain and realigns it to the output clock domain.

43.5.6 Low-level protocol (LLP)

The LPP module retrieves the raw data from the lane merger, and decodes the data flow respect to the CSI-2 low-level protocol. The software selects which data in the data interleaving flow need to be sent to the DCMIPP without being filtered out. The LLP is compliant with the MIPI CSI-2 specification as described in *Section 43.2*.

A data integrity check is done thanks to an ECC code for packet header error correction, and a 16-bit CRC checksum code for payload error detection. All these errors are flagged into specific registers, and may trig interruption for specific software action needs (see *Section 43.8* for more details).

Figure 367. Low-level protocol overview



The CSI-2 low-level protocol is mainly composed by short and long packets, each of them having a well identified role.

Long packet

The long packet format is presented in *Figure 368*. It contains a data ID that is composed of a virtual channel number for which the data packets are attached, and the corresponding data type. The LLP module filters data out by configuring CSI_VCxCFGRy registers (x = 0 to 3, y = 1 to 4), and VCxSTART/VCxSTOP bits in CSI_CR register.

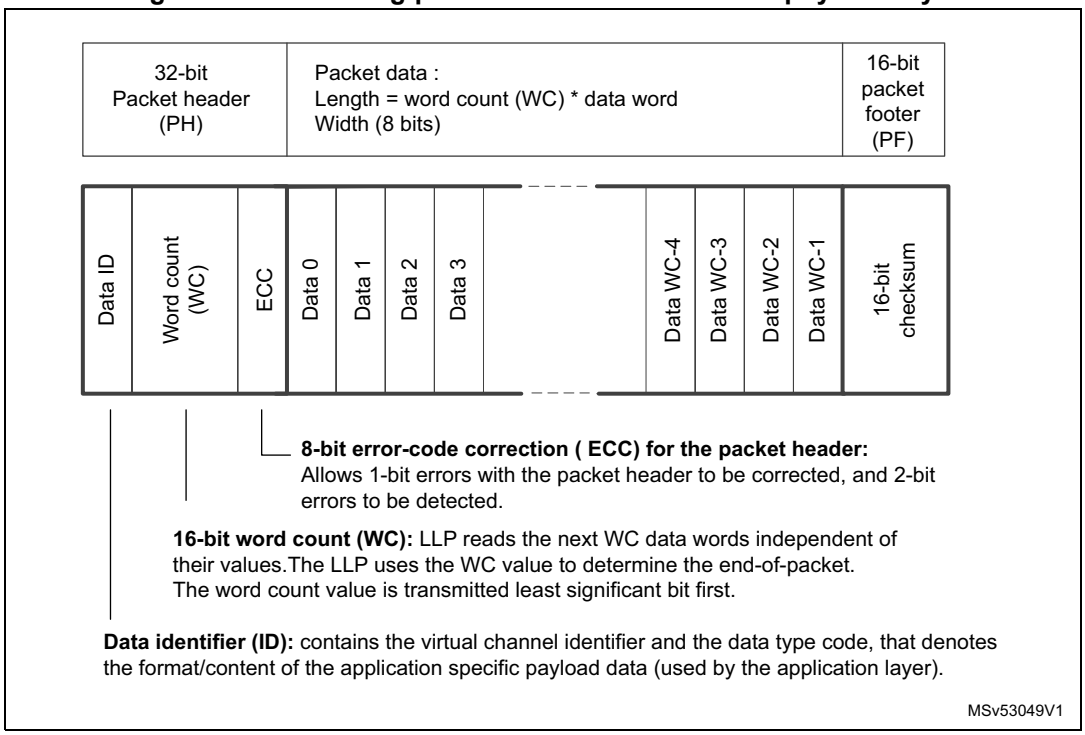
The word-count field allows the LLP to determine the data-payload width before detecting the packet footer and the EOT. This information is used to detect shorter-than-expected packets, and to trigger a corresponding error (refer to *Section 43.8*).

The ECC corrects a single erroneous bit in the packet header in the case of data corruption. An error is raised if there are two errors in total (the LLP decoder immediately stops the current frame), because only one bit can be corrected.

The packet data corresponds to the data payload for a given virtual channel and for an identified data type.

To complete the long-packet reception block, a packet footer that contains a 16-bit CRC checksum is received and compared to that which is internally calculated by the LLP, to check the data integrity of the received data payload.

Figure 368. CSI2- long-packet structure for a D-PHY physical layer

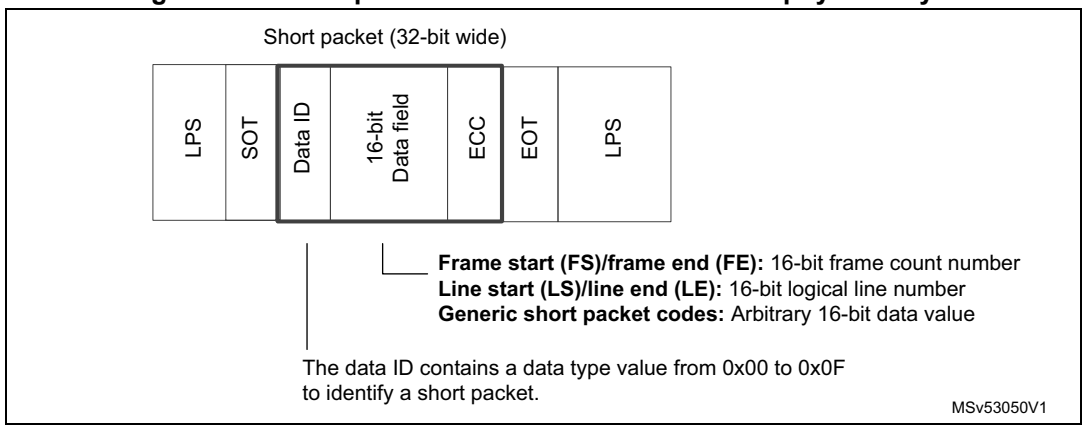


Short packet

The CSI-2 short-packet format for a D-PHY physical layer is shown in *Figure 369*.

It is similar to the long packet header, except that the word count field is replaced by a 16-bit data field. The ECC 8-bit code is present to secure headers having potential bit corruption.

Figure 369. Short-packet structure based on D-PHY physical layer

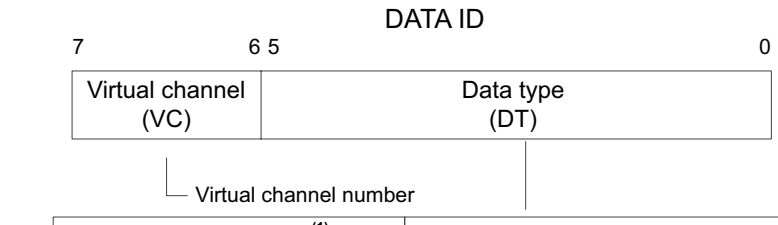


Data ID field

The 8-bit data ID field comprises 2 bits for the virtual channel number coding, and 6 bits to identify the packet data type (short or long packet), and the payload data packet format in the case of long packet reception.

The LLP filters subsequent packets depending on the user configuration indicated in the previously received data ID field (see *Figure 370*).

Figure 370. DATA ID structure in short or long packet format



Data type classes (1)	Description
0x00 to 0x07	Synchronization short packet data types
0x08 to 0x0F	Generic short packet data types
0x10 to 0x17	Generic long packet data types
0x18 to 0x1F	YUV data
0x20 to 0x27	RGB data
0x28 to 0x2F	RAW data
0x30 to 0x37	User defined byte-based data
0x38 to 0x3F	Reserved

(1) Refer to the MIPI Alliance specification for the CSI-2 to have the detail of each intermediate value not described in this table.

MSv53051V1

Data type field

The data type (DT) fields are configured by the application code to determine which data types are processed by the LLP for a given virtual channel. CSI_VCxCFGRy registers are used to configure the data types to be considered by the LLP.

Each virtual channel is managed independently, and up to seven different virtual-channel data types can be declared in parallel. All data type input to the LLP that are not defined for a given virtual channel are filtered out by the LLP module and discarded.

Data types in the video data flow are located:

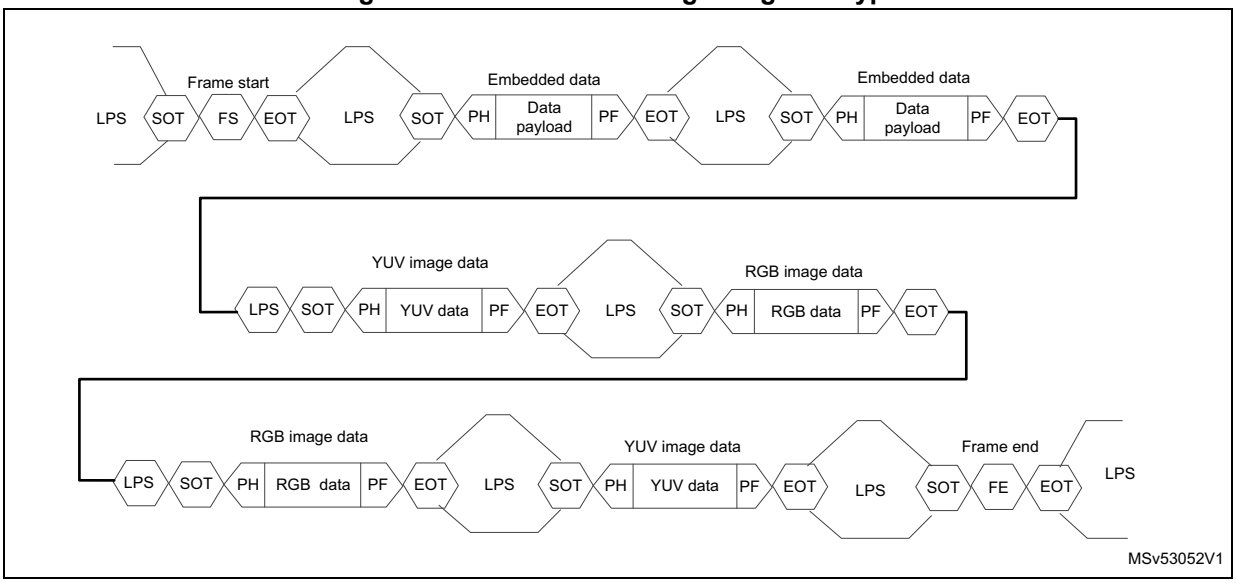
- in the data ID packet for a short packet
- in the packet header (PH) for a long packet format

Data interleaving: data-type based

In the video data flow, the transmitter can merge, on the same physical node, multi-device data transmissions having different data types. The LLP filters out these multi-device flows depending on the application configuration. A typical video data flow interleaving multiple data types is shown in *Figure 371*.

The application software must configure CSI_VCxCFGRy registers to enable data types for a virtual channel, and to set data-type formats to be processed. The LLP reorders the input data in a format that optimizes memory usage (see *Figure 371* to *Figure 379*).

Figure 371. Data interleaving using data type



Data interleaving: virtual channels

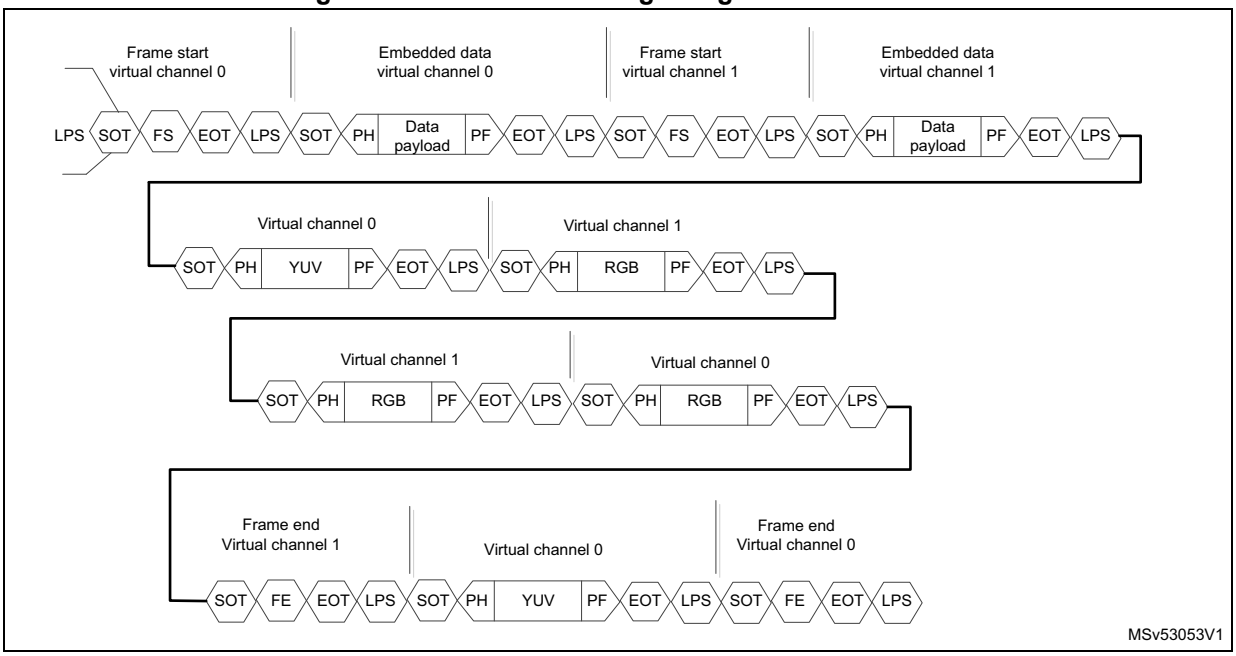
The data interleaving in the video data flow can also be managed by the virtual channel ID. Each external device can have a specific virtual channel identifier in order for the software layer or the DCMIPP to properly handle data packets belonging to a device.

The application code defines which of the virtual channels have to be considered by the LPP without any filtering operation. The CSI_CR register must be configured accordingly by activating the virtual channel (via VCxSTART), or by setting VCxSTOP to deactivate the virtual channel.

The virtual channel number is encapsulated in the data ID packet in short or long packet.

A typical data interleaving case is illustrated in *Figure 372*.

Figure 372. Data interleaving using virtual channels



Data formats

The LLP, in addition to a data-flow filter role, handles data reordering for memory optimization, storage, or post-processing purposes.

The CSI-2 Host controller is compliant with the MIPI Alliance CSI-2 specification. It mainly supports the following data formats:

- BPP6 (for example RAW6, RGB666)
- BPP7 (for example RAW7)
- BPP8 (for example RAW8, YUV 8-bits, RGB444, RGB555, RGB565, RGB888, JPEG)
- BPP10 (for example RAW10, YUV 10-bits)
- BPP12 (for example RAW12)
- BPP14 (for example RAW14)
- BPP16 (for example RAW16)

Figure 373. RAW6 (BPP6) input and output format

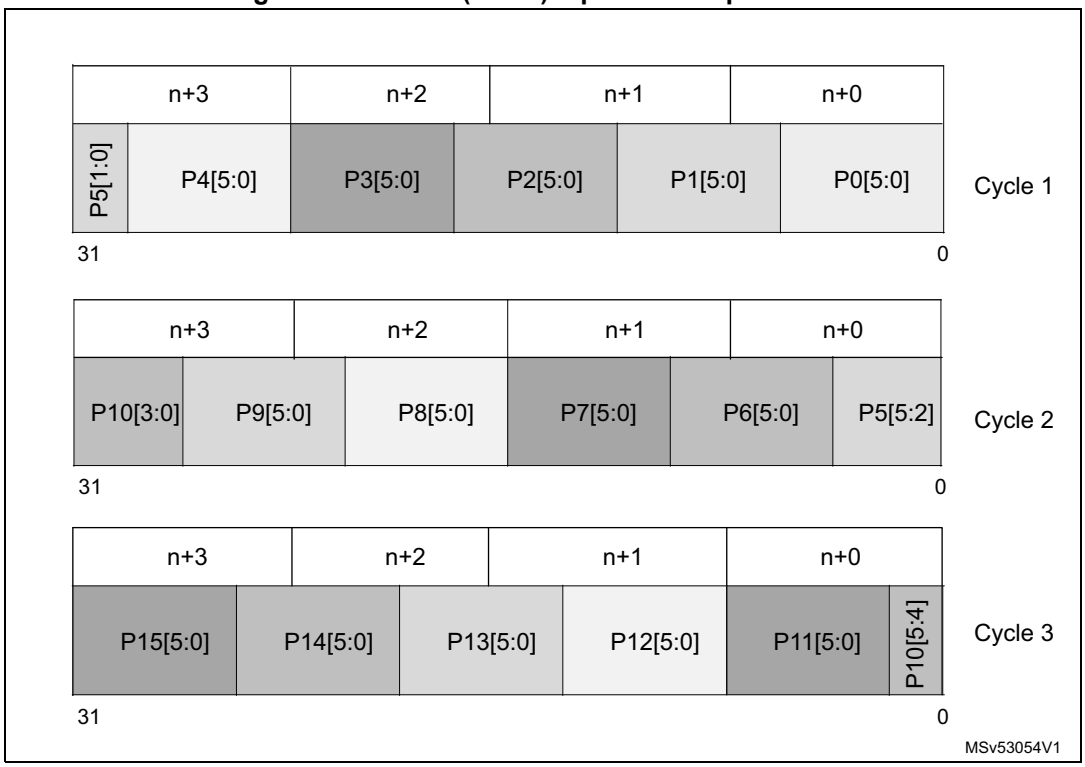


Figure 374. Raw7 (BPP7) input and output format

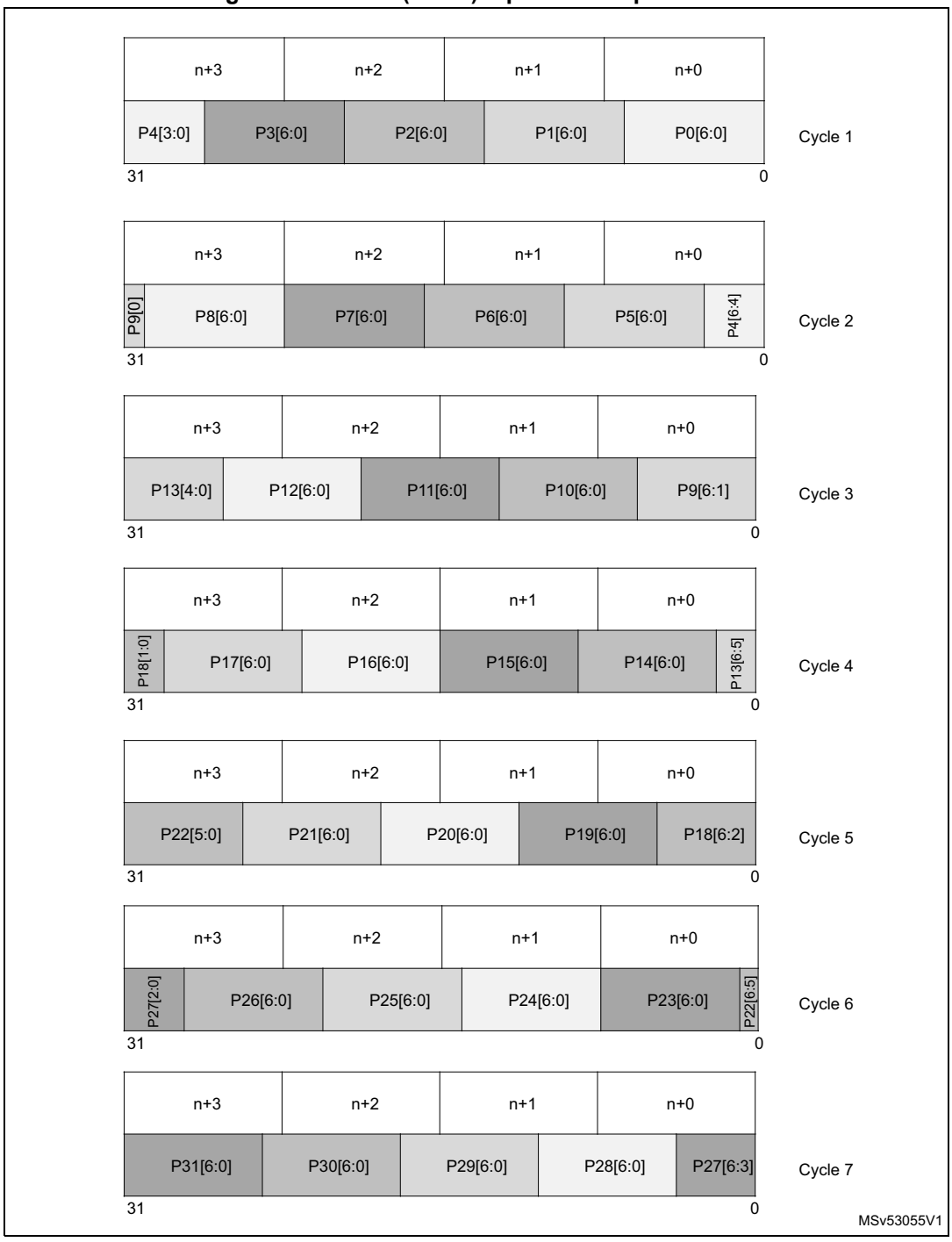


Figure 375. Raw8 (BPP8) input and output format

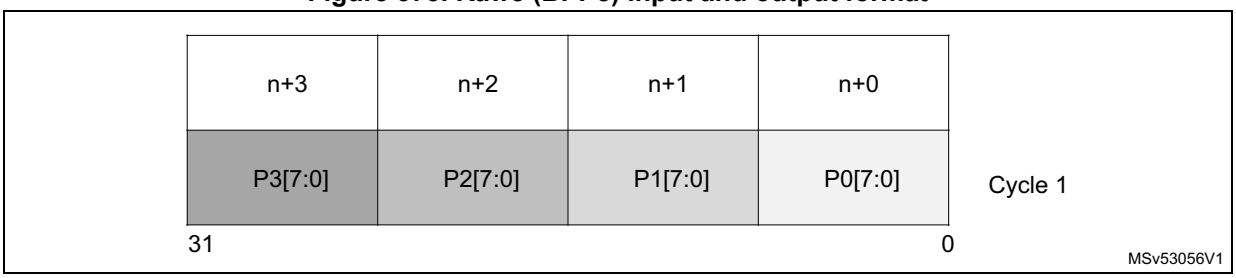


Figure 376. Raw 10 (BPP10) input and output format

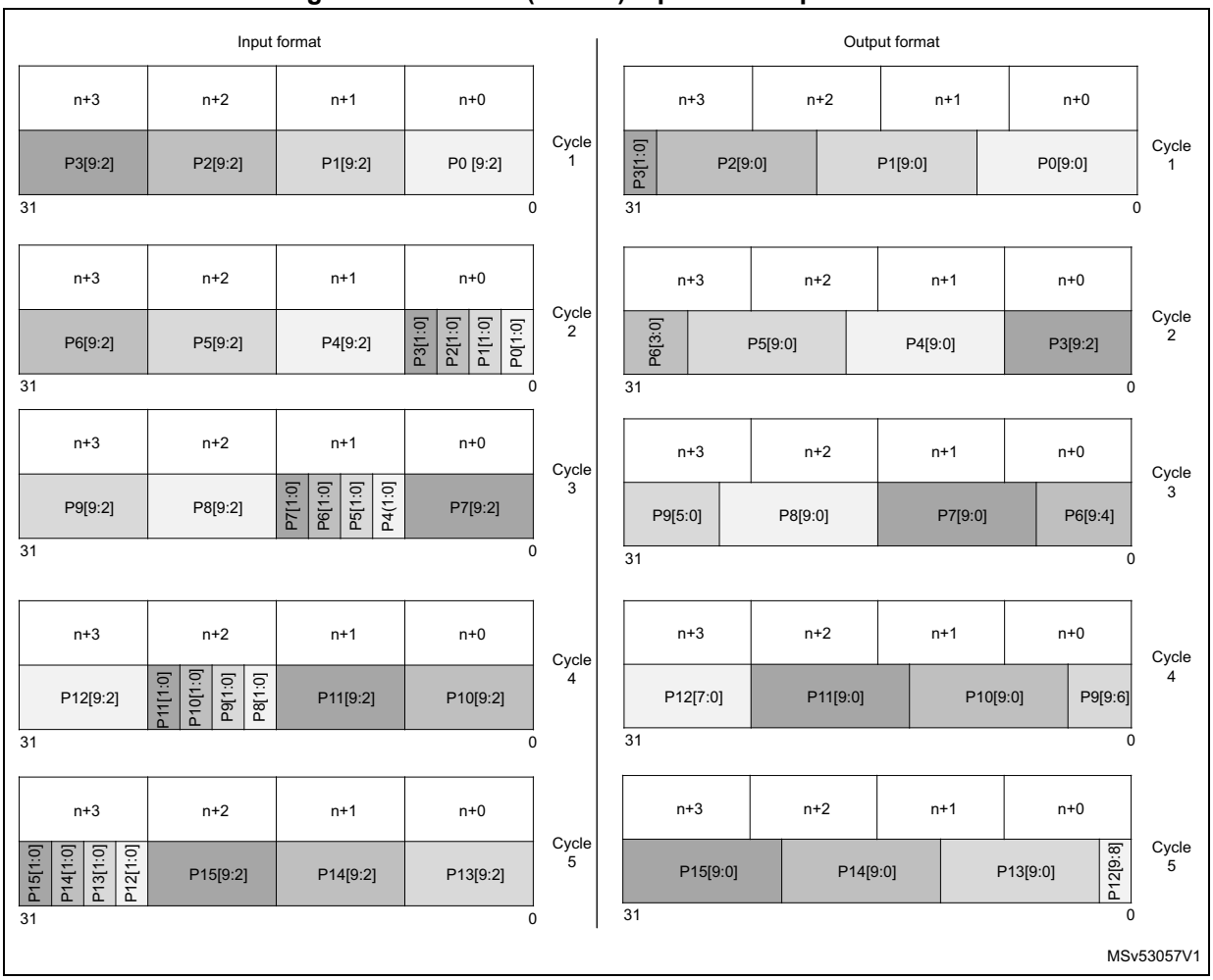


Figure 377. Raw12 (BPP12) input/output format

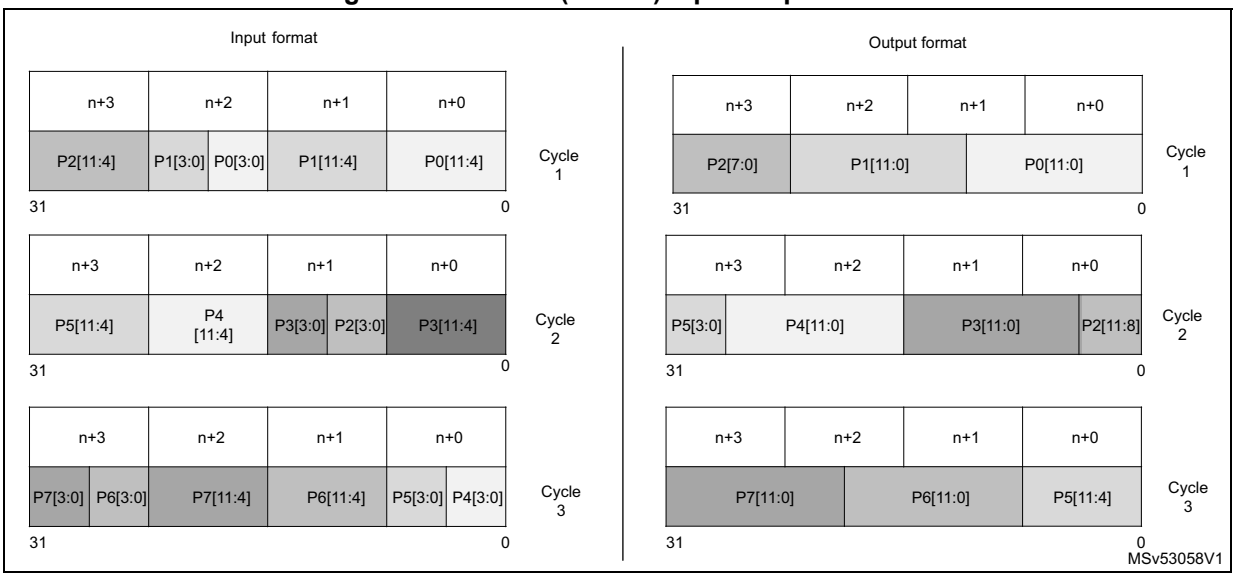


Figure 378. Raw14 (BPP14) input format

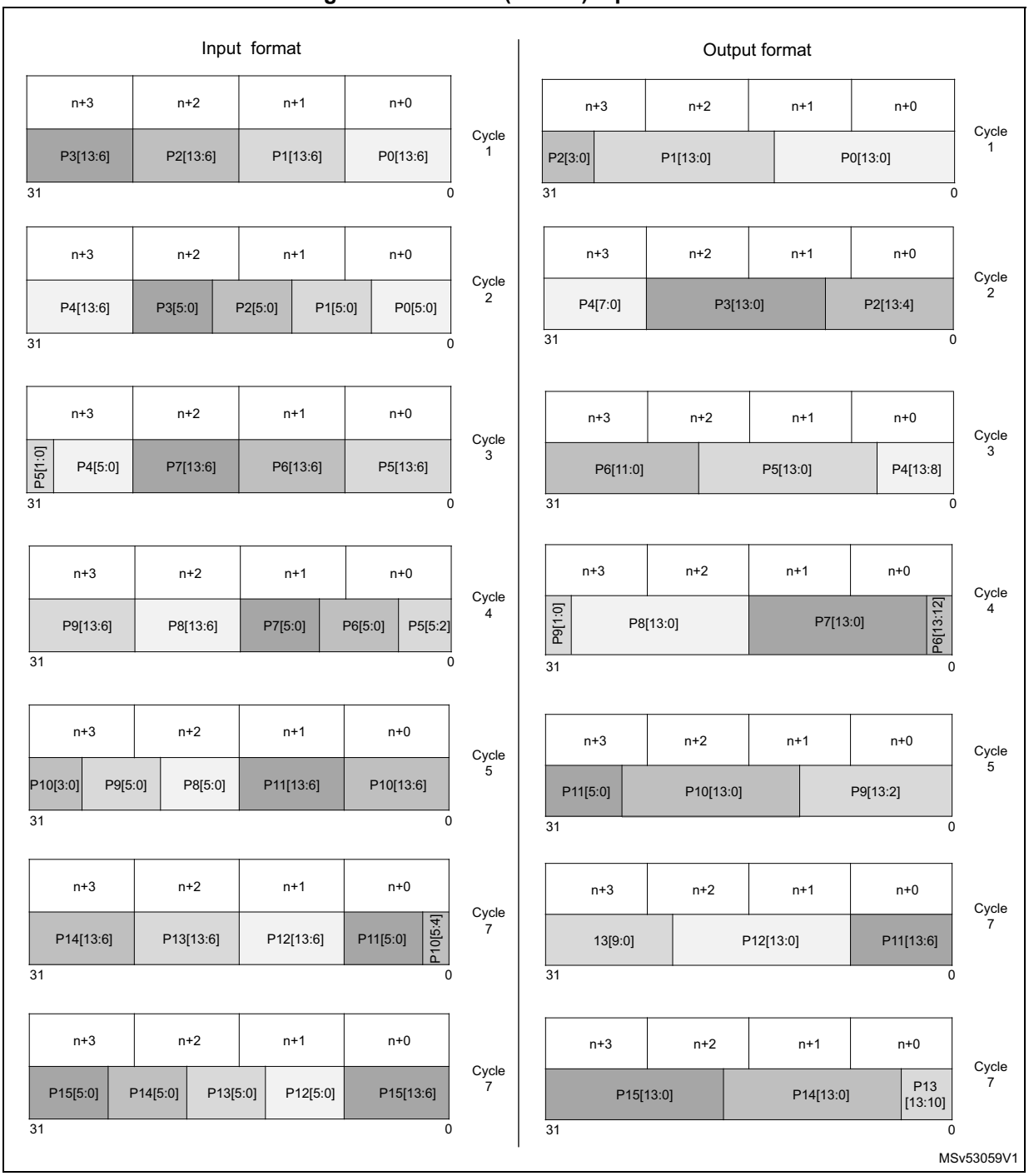
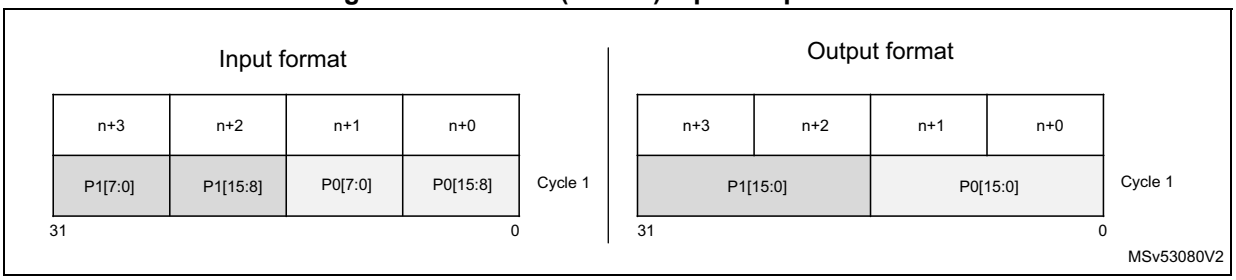


Figure 379. Raw16 (BPP16) input/output format



43.5.7 CSI-2 Host protocol output

The CSI-2 Host controller generates ISB-Byte header data according to CSI_VCxCFGRy. The ISB-Byte header generation respects the following rules:

- ISB-Byte header channel number is identical to the CSI-2 virtual channel and data type (LSB aligned).
- ISB-Byte header data type is identical to CSI-2 data type (LSB aligned).
- ISB-Byte header frame number is identical to the CSI-2 frame number.
- ISB-Byte header count is identical to CSI-2 word count.
- The format is updated according to CSI_VCxCFGRy configuration.

The only differences between the ISB-Byte data flow in input to the LLP and the ISB-Byte header bus at the output are listed below:

- There is no ECC information in the data corresponding to the short packet information.
- The packet footer (containing the CRC-16 value) is not transmitted, or forwarded on the output since the decoder already checked it from the input, and raised an error if the data payload is corrupted.
- The data payload output format is reordered and compared to the input data format of the LLP in order to ease pixels storage in memory for further hardware/software post-processing.
- Only identified virtual channels and their selected data types are generated on the output.

The ISB-Byte header is used by the DCMIPP peripheral to handle post-processing data from the external devices.

43.6 CSI-2 Host programming guide

43.6.1 CSI-2 PHY setup

The following actions are required to setup the CSI-2 PHY:

1. Quit the hardware reset state with PEN = 0x01 in CSI_PRCR.
2. Configure frequency settings with:
   – DLD = 0x00 in CSI_PFCR (to remain in Rx)
   – HSFR in CSI_PFCR that depends on the bit-rate expected on the data lanes
   – CCFR = RoundUp((F - 17) × 4) in CSI_PFCR (F = configuration clock frequency)
3. Configure internal registers via the test interface, with TRSEN = TCKEN = 0x0 in CSI_PTCR0.

43.6.2 CSI-2 Host controller setup and data reception start

The D-PHY RX software initialization is required to set up the CSI-2 Host controller.

Lane-merger configuration (static configuration)

The lane merger must be configured while CSIEN = 0 in CSI_CR, and when the D-PHY_RX is not streaming (when it is not providing a clock).

To map the physical data lane to the logical data lane, the lane merger needs to be configured through DL0MAP and DL1MAP in CSI_LMCFGR. In addition, the number of valid data lanes must be configured through LANENB in the same register.

LLP configuration

1. Configure the watchdog value in CSI_WDR. This corresponds to the maximum number of clock cycles between two packets before an input frame is considered erroneous and the transmission failed. It can be left at 0x0 to disable the watchdog feature.
2. Enable the CSI-2 Host controller with CSIEN = 1 in CSI_CR.
3. Configure the selected virtual channels in CSI_VCxCFGRy.

   If all data types must be captured with the same format, in CSI_VCxCFGR1:
   - Set ALLDT = 0x011.
   - Select the format with CDTFT[4:0].

   If only a few data types are to be captured, or if the format is not identical for all data types, then up to seven data types can be selected:
   - Set ALLDT = 0x0 in CSI_VCxCFGR1.
   - Set DTyEN = 0x1 (y = 0 to 6) in CSI_VCxCFGR1 for the data types used.
   - Set DTy[5:0] in CSI_VCxCFGRy to the selected data type.
   - Set DTyFT[4:0] in CSI_VCxCFGRy to the selected format.
4. Depending on application requirements, specific software actions may detect particular events in a frame, and trigger a corresponding interrupt, with the following:
   - Set LINECNT[15:0] and BYTECNT[15:0] in CSI_LBxCFGR to trigger an event when the data corresponding to a specific byte within a specific line is detected by the LLP. Up to four independent triggers can be programmed and linked to a selected virtual channel through LBxVC[1:0] in CSI_PRGITR. The selected trigger can be enabled by setting LBxEN in the same register.
   - Set COUNT[24:0] in CSI_TIMxCFGR to trigger an event after a specified number of clock cycles (clk_proc) from an SOF or an EO, depending on TIMxEOF and TIMxEN in CSI_PRGITR. Up to four independent timer triggers can be programmed and linked to a selected virtual channel through TIMxVC[1:0] in CSI_PRGITR.
5. Enable interrupt sources for which the code needs to make specific actions, by configuring CSI_IER0 and CSI_IER1 registers.
6. D-PHY_RX lanes can be enabled by setting DL0EN and/or DL1EN, and CLEN in CSI_PCR.
7. Start the virtual channel by setting VCxSTART = 0x1 in CSI_CR.

43.6.3 Data reception stop

To stop data reception, set VCxSTOP = 1 in CSI_CR. The data reception stops at the end of the current frame. The virtual channel status can be retrieved from VCxSTATEF in CSI_SR0.

Note: A virtual channel cannot start before it has been completely stopped (VCxSTATEF = 0 in CSI_SR0).

43.7 CSI-2 Host low-power modes

Table 415. Effect of low-power modes on CSI-2 Host controller

Mode	Description
Sleep	No effect. The CSI-2 host interrupts cause the device to exit the Sleep mode.
Stop	The CSI-2 host is in power-down, and must be reinitialized after exiting Stop mode.
Standby	The CSI-2 host is in power-down, and must be reinitialized after exiting Standby mode.

43.8 CSI-2 Host interrupts and errors

Few events can generate an error flag or an interrupt from the CSI-2 Host controller interrupt handler.

The table below defines the error sources flagged by the CSI-2 HOST controller.
The content of CSI_ERR1 and CSI_ERR2 registers can be used to know exactly from which virtual number and data types a particular error appeared and has been detected. It can be used as debug purpose too.

Table 416. List of errors detected by the CSI-2 Host controller

Error description	Bit name	Register
Virtual channel number on which a data ID information error is detected	IDVCERR[1:0]	CSI_ERR1
Data type for which a data ID information error is detected	IDDTERR[5:0]	
Virtual channel number for which the ECC has corrected one bit in error	CECCVCERR[1:0]	
Data type for which the ECC has corrected one bit in error	CECCDTERR[5:0]	
Virtual channel number for which a CRC error is detected	CRCVCERR[1:0]	
Data type for which a CRC error is detected	CRCDTERR[5:0]	
Virtual channel number on which the invalid synchronization has been detected	SYNCVCERR[1:0]	CSI_ERR2
Virtual channel number on which a watchdog error has been detected	WDVCERR[1:0]	
Virtual channel number on which a shorter packet than expected has been detected	SPKTVCERR[1:0]	
Data type for which the shorter packet than expected error has been detected	SPKTDTERR[5:0]	

The table below shows the interrupt sources that are triggered by the CSI-2 Host controller if the corresponding bit is enabled.

Table 417. List of events that trigger an interrupt

Interrupt flag	Register	Interrupt sources
SYNCERRF	CSI_SR0	Synchronization error. The interrupt is enabled if SYNCERRIE is set in CSI_IER0(1).
WDERRF		Watchdog error. The interrupt is enabled if WDERRIE is set in CSI_IER0(1).
SPKTERRF		Shorter packet than expected error. The interrupt is enabled if SPKTERRIE is set in CSI_IER0 register(1).
IDERRF		Data ID information error. The interrupt is enabled if IDERRIE is set in CSI_IER0(1).
CECCERRF		Corrected ECC error. The interrupt is enabled if CECCERRIE is set in CSI_IER0(1).
ECCERRF		ECC error. The interrupt is enabled if ECCERRIE is set in CSI_IER0(1).
CRCERRF		CRC error. The interrupt is enabled if CRCERRIE is set in CSI_IER0(1).
CCFIFOFF		Clock changer FIFO full event. The interrupt is enabled if CCFIFOFFIE is set in CSI_IER0(2).
SPKTF		Short packet detection. The interrupt is enabled if SPKTIE is set in CSI_IER0(2).
EOF3F		EOF on virtual channel 3. The interrupt is enabled if EOF3IE is set in CSI_IER0(2).
EOF2F		EOF on virtual channel 2. The interrupt is enabled if EOF2IE is set in CSI_IER0(2).
EOF1F		EOF on virtual channel 1. The interrupt is enabled if EOF1IE is set in CSI_IER0(2).
EOF0F		EOF on virtual channel 0. The interrupt is enabled if EOF0IE is set in CSI_IER0(2).
SOF3F		SOF on virtual channel 3. The interrupt is enabled if SOF3IE is set in CSI_IER0(2).
SOF2F		SOF on virtual channel 2. The interrupt is enabled if SOF2IE is set in CSI_IER0(2).
SOF1F		SOF on virtual channel 1. The interrupt is enabled if SOF1IE is set in CSI_IER0(2).
SOF0F		SOF on virtual channel 0. The interrupt is enabled if SOF0IE is set in CSI_IER0(2).
TIM3F		Timer 3 counter value reached. The interrupt is enabled if TIM3IE is set in CSI_IER0(2).
TIM2F		Timer 2 counter value reached. The interrupt is enabled if TIM2IE is set in CSI_IER0(2).
TIM1F		Timer 1 counter value reached. The interrupt is enabled if TIM1IE is set in CSI_IER0(2).
TIM0F		Timer 0 counter value reached. The interrupt is enabled if TIM0IE is set in CSI_IER0(2).
LB3F		Line/byte event 3 reached. The interrupt is enabled if LB3IE is set in CSI_IER0(2).
LB2F		Line/byte event 2 reached. The interrupt is enabled if LB2IE is set in CSI_IER0(2).
LB1F		Line/byte event 1 reached. The interrupt is enabled if LB1IE is set in CSI_IER0(2).
LB0F		Line/byte event 0 reached. The interrupt is enabled if LB0IE is set in CSI_IER0(2).

Table 417. List of events that trigger an interrupt (continued)

Interrupt flag	Register	Interrupt sources
ECTRLDL1F	CSI_SR1	Error control on data line 1. The interrupt is enabled if ECTRLDL1IE is set in CSI_IER1.
ESYNCESCDL1F		Low-power data transmission synchronization error on data line 1. The interrupt is enabled if ESYNCESCDL1IE is set in CSI_IER1.
EESCDL1F		Escape entry error on data line 1. The interrupt is enabled if EESCDL1IE is set in CSI_IER1.
ESOTSYNCDL1F		SOT synchronization error on data line 1. The interrupt is enabled if ESOTSYNCDL1IE is set inCSI_IER1.
ESOTDL1F		SOT error on data line 1. The interrupt is enabled if the bit ESOTDL1IE is set into CSI_IER1.
ECTRLDL0F		Error control on data line 0. The interrupt is enabled if the bit ECTRLDL0IE is set into CSI_IER1.
ESYNCESCDL0F		Low-power data transmission synchronization error on data line 0. The interrupt is enabled if the bit ESYNCESCDL0IE is set into CSI_IER1.
EESCDL0F		Escape entry error on data line 0. The interrupt is enabled if EESCDL0IE is set in CSI_IER1.
ESOTSYNCDL0F		SOT synchronization error on data line 0. The interrupt is enabled if ESOTSYNCDL0IE is set in CSI_IER1.
ESOTDL0F		SOT error on data line 0. The interrupt is enabled if ESOTDL0IE is set in CSI_IER1.

1. Refer to *Section 43.8.1: Error handling*.
2. Refer to *Section 43.8.2: Interrupt handling*.

43.8.1 Error handling

Details of the errors are given in CSI_ERR1 and CSI_ERR2.

Note: Except for data ID information, all other errors can occur during the product life, and must be handled by the software.

Corrected ECC error

When this error is detected, the CECCERR interrupt is triggered if CECCERRIE = 1 in CSI_IER0. This is an information event rather than a real error, as the CSI-2 Host controller successfully recovered the ECC error, so the frame is processed properly. Some details of the error are located in CECCVCERR and CECCDTERR in CSI_ERR1:

ECC error

When a multi-bit ECC error is detected, the ECCERR interrupt is triggered if ECCERRIE = 1 in CSI_IER0. As all virtual channels are potentially corrupted, the LPP immediately stops the current frames.

The CSI-2 Host controller restarts each enabled virtual channel on the next SOF packet of each channel. The software must perform the necessary error management (such as resetting some peripherals after this error detection). No details are available for the error as the relevant information is corrupted.

CRC error

When a CRC error is detected, the CRCERR interrupt is triggered if CRCERRIE = 1 in CSI_IER0. As the frame envelope is not corrupted, the CSI-2 Host controller does not stop the frame processing. The software must perform the necessary error management (such as reseting some peripherals, possibly including the CSI-2 Host controller). Details are available for this error in CRCVCERR and CRCDTERR in CSI_ERR1.

Data ID information

When a reserved data type is detected, the IDERR interrupt is triggered if IDERRIE = 1 in CSI_IER0. This is an information event rather than a real error, as the hardware processes this packet as for any non-erroneous packet.Details for the error are available in IDVCERR and IDDTERR in CSI_ERR1.

Invalid synchronization

When an invalid synchronization is detected, the SYNCERR interrupt is triggered if SYNCERRIE = 1 in CSI_IER0. An invalid synchronization is either an SOF short packet during a frame, or an EOF short packet during an inter-frame. It is an information event rather than a real error, as the hardware is able to handle such an error:

- An SOF short packet during a frame is interpreted as back-to-back EOF and SOF. It behaves accordingly (ISB-Byte header EOF, ISB-Byte header SOF, and so on). The SOFx interrupt is generated in addition to the SYNCERR one if SOFxIE = 1 in CSI_IER0.
- An EOF short packet during an inter-frame is ignored (no EOFx interrupt is generated whatever the state of EOFx in CSI_IER0). Details for the error are available in SYNCVCERR in CSI_ERR2.

Shorter than expected packet

When a shorter-than-expected packet is received, the SPKTERR interrupt is triggered if SPKTERRIE = 1 in CSI_IER0. The packet is padded up to its theoretical size, and the frame is processed as if no error occurred. The software must perform the necessary error management (such as reseting some peripherals, possibly including the CSI decoder). Details for the error are available in SPKTVCERR and SPKTDTERR in CSI_ERR2.

Watchdog

When the time between two packets of a given virtual channel is longer than the programmed watchdog (CNT[31:0] in CSI_WDR), the WDERR interrupt is triggered if WDERRIE = 1 in CSI_WDR. As the corresponding virtual channel stream is corrupted, the CSI2 Host controller immediately stops processing this virtual channel. The software must perform the necessary error management (such as reseting some peripherals after this error detection), before restarting the CSI-2 Host. Details for the error are available in WDVCERR in CSI_ERR2.

43.8.2 Interrupt handling

The CSI-2 Host controller offers many interrupt sources that can be activated if the associated enable bit is set. Some interrupt sources are linked to an error detection from the D-PHY physical layer or the CSI-2 Host controller (refer to *Section 43.8.1*).

Short packet interrupt

This interrupt notifies the software that a generic short packet has been received. The software can retrieve the associated data in CSI_SPDFR. This register value is overwritten with each new generic short packet.

SOF and EOF interrupts

The CSI-2 Host controller generates an SOF interrupt for each SOF short packet received. It also generates an EOF interrupt for each EOF short packet, independently for all four virtual channels. These interrupts are generated even if the virtual channel is disabled. The software must mask or ignore these interrupts if they are not needed.

Line/byte programmable interrupts

The CSI-2 Host controller provides four interrupts that can be programmed to be triggered at any point during the active data of a frame. In order to set a programmable line/byte interrupt, the software must follow these steps:

1. Select a virtual channel by setting LBxVC[1:0] in CSI_PRGITR.
2. Select the frame location at which the interrupt is triggered by setting LINECNT and BYTECNT in CSI_LBxCFGRy.
3. Enable the line/byte counting by setting LBxEN in the CSI_PGRITR register. This bit must be set only when the CSI-2 is enabled.
4. Enable the interrupt by setting LBxIE in CSI_IER0.
5. The interrupt is generated starting from the next frame on the selected virtual channel.

In order to stop a programmable interrupt, the software must reset the LBxIE bit in the CSI_IER0 register. It immediately stops the interrupt.

Note: All four line/byte programmable interrupts are independent.

Timer-based programmable interrupts

The CSI-2 Host controller provides four interrupts that can be programmed to be triggered based on a timer. In order to set a programmable timer based interrupt, the software must select a virtual channel by setting TIMxVC[1:0] in CSI_PRGITR. The following steps must be followed, while the CSI-2 is disabled:

1. Select the starting event of the timer with TIMxEOF in CSI_PRGITR.
2. Select the timer duration by setting COUNT[24:0] in CSI_TIMxCFGR.
3. Select and enable counting by setting TIMxEN in CSI_PRGITR.
4. Enable the interrupt by setting TIMxIE in CSI_IER0.
5. The interrupt is generated starting from the next frame on the selected virtual channel.

In order to stop a programmable interrupt, the software must reset the TIMxIE bit in the CSI_IER0 register. It immediately stops the interrupt.

Note: All four timer-based programmable interrupts are independent.

If the timer is set to a value longer than the frame period, the interrupt may not be generated for all frames.

Clock-changer FIFO full interrupt

The clock changer is also built around a FIFO which is never full in normal conditions. If the overall system works very slowly (for example due to the CSI-2 and DCMIPP processing clock frequency being set to an inadequate frequency, system-bus bandwidth availability, and so on), the clock-changer FIFO may become full. In this case, an interrupt is set if CCFIFOFIE = 1 in CSI_IER0.

43.9 CSI-2 Host registers

43.9.1 CSI-2 Host control register (CSI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VC3 STOP	VC3 START	Res.	Res.	VC2 STOP	VC2 START	Res.	Res.	VC1 STOP	VC1 START	Res.	Res.	VC0 STOP	VC0 START	Res.	CSIEN
w	w			w	w			w	w			w	w		rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **VC3STOP**: Virtual channel 3 stop

The CSI-2 decoder stops the virtual channel 3.

0: No effect

1: Pulse generation to stop the virtual channel 3 processing

Note: The CSI-2 decoder stops processing the virtual channel 3 after the next EOF event (or immediately if the Stop command is issued during frame blanking).

A write operation into this bit must not be simultaneous with a write into VC3START.

Reading this bit returns 0.

Bit 14 **VC3START**: Virtual channel 3 start

The CSI-2 decoder processes the virtual channel 3 according to CSI_VC3CFGRx (x = 0..3).

0: No effect

1: Pulse generation to start the virtual channel 3 processing

Note: The CSI-2 decoder starts processing the virtual channel 3 after the next SOF event.

A write operation into this bit must not be simultaneous with a write into VC3STOP.

Reading this bit returns 0.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **VC2STOP**: Virtual channel 2 stop
The CSI-2 decoder stops the virtual channel 2.
0: No effect
1: Pulse generation to stop the virtual channel 2 processing
Note: The CSI-2 decoder stops processing the virtual channel 2 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC2START.
Reading this bit returns 0.

Bit 10 **VC2START**: Virtual channel 2 start
The CSI-2 decoder processes the virtual channel 2 according to CSI_VC2CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 2 processing
Note: The CSI-2 decoder starts processing the virtual channel 2 after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC2STOP.
Reading this bit returns 0.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **VC1STOP**: Virtual channel 1 stop
The CSI-2 decoder stops the virtual channel 1.
0: No effect
1: Pulse generation to stop the virtual channel 1 processing
Note: The CSI-2 decoder stops processing the virtual channel 1 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC1START.
Reading this bit returns 0.

Bit 6 **VC1START**: Virtual channel 1 start
The CSI-2 decoder processes the virtual channel 1 according to CSI_VC1CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 1 processing
Note: The CSI-2 decoder starts processing the virtual channel 1 after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC1STOP.
Reading this bit returns 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **VC0STOP**: Virtual channel 0 stop
The CSI-2 decoder stops the virtual channel 0.
0: No effect
1: Pulse generation to stop the virtual channel 0 processing
Note: The CSI-2 decoder stops processing the virtual channel 0 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC0START.
Reading this bit returns 0.

Bit 2 **VC0START**: Virtual channel 0 start
he CSI-2 decoder processes the virtual channel 0 according to CSI_VC 0CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 0 processing
Note: The CSI-2 decoder starts processing the virtual channel 0after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC0STOP.
Reading this bit returns 0.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CSIEN**: CSI-2 enable

This bit enables the CSI-2 peripheral for processing the data flow coming from the camera through the D-PHY_RX.

0: CSI-2 disabled

1: CSI-2 enabled

43.9.2 CSI-2 Host DPHY_RX control register (CSI_PCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1EN	DL0EN	CLEN	PWR DOWN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **DL1EN**: D-PHY_RX data lane 1 enable

This bit enables the data lane 1 for the D-PHY_RX.

0: Data lane 1 module is in shutdown mode.

1: Enable the data lane 1 module.

Bit 2 **DL0EN**: D-PHY_RX data lane 0 enable

This bit enables the data lane 0 for the D-PHY_RX.

0: Data lane 0 module is in shutdown mode.

1: Enable the data lane 0 module.

Bit 1 **CLEN**: Clock lane enable

This bit enables the clock lane module of the D-PHY_RX.

0: Clock lanes disabled

1: Clock lanes enabled

Bit 0 **PWRDOWN**: Power down

This bit keeps the D-PHY_RX in power-down state.

0: Active state

1: Power down the D-PHY_RX

43.9.3 CSI-2 Host virtual channel x configuration register 1 (CSI_VCxCFGR1)

Address offset: 0x10 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT0FT[4:0]					Res.	Res.	DT0[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CDTFT[4:0]					DT6EN	DT5EN	DT4EN	DT3EN	DT2EN	DT1EN	DT0EN	ALLDT
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT0FT[4:0]**: Data type 0 format

These bits configure the input pixel format for the virtual channel x on data type 0.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: reserved and must not be used

Note: The format selection is valid only if ALLDT = 0 and DT0EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT0[5:0]**: Data type 0 class selection for virtual channel x

These bits select the data type 0 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **CDTFT[4:0]**: Common format for all data types

These bits are configured to select which input pixel format is applied for all the seven data type on virtual channel x.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved

Note: The format selection is valid only if ALLDT = 1. These bits have no effect if ALLDT = 0.

Bit 7 **DT6EN**: Data type 6 enable

This bit enables the data type 6 for the virtual channel x. The data type format is applied with respect to the configuration of DT6[5:0] and DT6FT[4:0] in CSI_VCxCFGR4.

0: Data type 6 for virtual channel x is disabled.

1: Data type 6 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 6 **DT5EN**: Data type 5 enable

This bit enables the data type 5 for the virtual channel x. The data type format is applied with respect to the configuration of DT5[5:0] and DT5FT[4:0] in CSI_VCxCFGR4.

0: Data type 5 for virtual channel x is disabled.

1: Data type 5 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 5 **DT4EN**: Data type 4 enable

This bit enables the data type 4 for the virtual channel x. The data type format is applied with respect to the configuration of DT4[5:0] and DT4FT[4:0] in CSI_VCxCFGR3.

0: Data type 4 for virtual channel x is disabled.

1: Data type 4 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 4 **DT3EN**: Data type 3 enable

This bit enables the data type 3 for the virtual channel x. The data type format is applied with respect to the configuration of DT3[5:0] and DT3FT[4:0] in CSI_VCxCFGR3.

0: Data type 3 for virtual channel x is disabled.

1: Data type 3 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 3 **DT2EN**: Data type 2 enable

This bit enables the data type 2 for the virtual channel x. The data type format is applied with respect to the configuration of DT2[5:0] and DT2FT[4:0] in CSI_VCxCFGR2.

0: Data type 2 for virtual channel x is disabled.

1: Data type 2 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 2 **DT1EN**: Data type 1 enable

This bit enables the data type 1 for the virtual channel x. The data type format is applied with respect to the configuration of DT1[5:0] and DT1FT[4:0] in CSI_VCxCFGR2.

0: Data type 1 for virtual channel x is disabled.

1: Data type 1 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 1 **DT0EN**: Data type 0 enable

This bit enables the data type 0 for the virtual channel x. The data type format is applied with respect to the configuration of DT0[5:0] and DT0FT[4:0] in CSI_VCxCFGR1.

0: Data type 0 for virtual channel x is disabled.

1: Data type 0 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 0 **ALLDT**: All data types enable for the virtual channel x

This bit allows to capture all data types according to CDTFT[4:0] for the virtual channel x.

0: Data type capture managed individually

1: All data types captured for virtual channel x

43.9.4 CSI-2 Host virtual channel x configuration register 2 (CSI_VCxCFGR2)

Address offset: 0x14 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT2FT[4:0]					Res.	Res.	DT2[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT1FT[4:0]					Res.	Res.	DT1[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT2FT[4:0]**: Data type 2 format

These bits configure the input pixel format for the virtual channel x on data type 2.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT2EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT2[5:0]**: Data type 2 class selection for virtual channel x

These bits select the data type 2 class and subclass for the virtual channel x according to the MIPI specification for CSI- 2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT1FT[4:0]**: Data type 1 format

These bits configure the input pixel format for the virtual channel x on data type 1.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7 -it words (for example RAW7)
00010: (BPP8) 8 -it words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12 -it words (for example RAW12)
00101: (BPP14) 14 -it words (for example RAW14)
00110: (BPP16) 16 -it words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT1EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT1[5:0]**: Data type 1 class selection for virtual channel x

These bits select the data type 1 class and subclass for the virtual channel x according to the MIPI specification for CSI- 2.

43.9.5 CSI-2 Host virtual channel x configuration register 3 (CSI_VCxCFGR3)

Address offset: 0x18 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT4FT[4:0]					Res.	Res.	DT4[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT3FT[4:0]					Res.	Res.	DT3[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT4FT[4:0]**: Data type 4 format

These bits configure the input pixel format for the virtual channel x on data type 4.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT4EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT4[5:0]**: Data type 4 class selection for virtual channel x

These bits select the data type 4 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT3FT[4:0]**: Data type 3 format

These bits configure the input pixel format for the virtual channel x on data type 3.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT3EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT3[5:0]**: Data type 3 class selection for virtual channel x

These bits select the data type 3 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

43.9.6 CSI-2 Host virtual channel x configuration register 4 (CSI_VCxCFGR4)

Address offset: 0x1C + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT6FT[4:0]					Res.	Res.	DT6[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT5FT[4:0]					Res.	Res.	DT5[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT6FT[4:0]**: Data type 6 format

These bits configure the input pixel format for the virtual channel x on data type 6.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT6EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT6[5:0]**: Data type 6 class selection for virtual channel x

These bits select the data type 6 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT5FT[4:0]**: Data type 5 format

These bits configure the input pixel format for the virtual channel x on data type 5.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved.

Note: The format selection is valid only if ALLDT = 0 and DT5EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT5[5:0]**: Data type 5 class selection for virtual channel x

These bits select the data type 5 class and subclass for the virtual channel x according to the MIPI specification for Camera Serial Interface 2.

43.9.7 CSI-2 Host line byte x configuration register (CSI_LBxCFGR)

Address offset: 0x50 + 0x04 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while LBxEN bit is set into the CSI_PRGITR is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LINECNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTECNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **LINECNT[15:0]**: Line counter

These bits select the line number, associated to LINECNT[15:0], to trigger, if enabled, the corresponding line/byte interruption flag based on the selected virtual channel identified in CSI_PRGITR.

Bits 15:0 **BYTECNT[15:0]**: Byte counter

These bits select the byte number, associated to BYTECNT[15:0], to trigger, if enabled, the corresponding line/byte interruption flag based on the selected virtual channel identified in CSI_PRGITR.

43.9.8 CSI-2 Host timer x configuration register (CSI_TIMxCFGR)

Address offset: 0x60 + 0x04 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **COUNT[24:0]**: Clock cycle counter

This register is used to specify how many clock cycles (clk_proc) the timer needs to count from a SOF on the considered virtual channel, before triggering the associated interrupt flag if enabled in CSI_PRGITR. The interruption is generated if TIMxIE is set in CSI_IER0.

43.9.9 CSI-2 Host lane merger configuration register (CSI_LMCFGR)

Address offset: 0x70

Reset value: 0x4321 0200

This register must not be modified while the CSI-2 is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1MAP[2:0]			Res.	DL0MAP[2:0]		
									rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LANENB[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw								

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **DL1MAP[2:0]**: Physical mapping of logical data lane 1

These bits are used to map the physical data lane 1 on the logical data lane.
001: Physical data lane 1 connected to logical data lane 0
010: Physical data lane 1 connected to logical data lane 1
Others: Reserved

Note: This field value must be different from DL0MAP[2:0].

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DL0MAP[2:0]**: Physical mapping of logical data lane 0

These bits are used to map the physical data lane 0 on the logical data lane.
001: Physical data lane 0 connected to logical data lane 0
010: Physical data lane 0 connected to logical data lane 1
Others: Reserved

Note: This field value must be different from DL1MAP[2:0].

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **LANENB[2:0]**: Number of lanes

These bits are used to specify the number of lanes used for the data reception.
001: 1 lane for the reception
010: 2 lanes for the reception
Others: Reserved

Bits 7:0 Reserved, must be kept at reset value.

43.9.10 CSI-2 Host program interrupt register (CSI_PRGITR)

Address offset: 0x74

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM3 EN	TIM3 EOF	TIM3VC[1:0]		TIM2 EN	TIM2 EOF	TIM2VC[1:0]		TIM1 EN	TIM1 EOF	TIM1VC[1:0]		TIM0 EN	TIM0 EOF	TIM0VC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LB3 EN	Res.	LB3VC[1:0]		LB2 EN	Res.	LB2VC[1:0]		LB1 EN	Res.	LB1VC[1:0]		LB0 EN	Res.	LB0VC[1:0]	
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 31 **TIM3EN**: TIM3 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM3EOF. An interrupt is generated if TIM3IE is set int CSI_IER0.
0: Timer 3 clock cycle counter not active
1: Timer 3 clock cycle counter active

Note: Enabling this bit locks TIM3EOF and TIM3VC[1:0] values.
The counting starts from the next frame after setting this bit.
This bit must be written only when the CSI-2 is disabled.

Bit 30 **TIM3EOF**: TIM3 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.
0: SOF selected as starting point for the counter
1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM3EN = 1 in CSI_PRGITR.
This bit must be written only when the CSI-2 is disabled.

Bits 29:28 **TIM3VC[1:0]**: TIM3 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM3 counter is linked.
00: Virtual channel 0 used to generate the TIM3 event
01: Virtual channel 1 used to generate the TIM3 event
10: Virtual channel 2 used to generate the TIM3 event
11: Virtual channel 3 used to generate the TIM3 event

Note: The write access to these bits is ignored if TIM3EN = 1 in CSI_PRGITR.
These bits must be written only when the CSI-2 is disabled.

Bit 27 **TIM2EN**: TIM2 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM2EOF. An interrupt is generated if TIM2IE is set in CSI_IER0.

0: Timer 2 clock cycle counter not active.

1: Timer 2 clock cycle counter active

Note: Enabling this bit locks TIM2EOF and TIM2VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 26 **TIM2EOF**: TIM2 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM2EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 25:24 **TIM2VC[1:0]**: TIM2 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM2 counter is linked.

00: Virtual channel 0 used to generate the TIM2 event

01: Virtual channel 1 used to generate the TIM2 event

10: Virtual channel 2 used to generate the TIM2 event

11: Virtual channel 3 used to generate the TIM2 event

Note: The write access to these bits is ignored if TIM2EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 23 **TIM1EN**: TIM1 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM1EOF. An interrupt is generated if TIM1IE is set in CSI_IER0.

0: Timer 1 clock cycle counter not active.

1: Timer 1 clock cycle counter active

Note: Enabling this bit locks TIM1EOF and TIM1VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 22 **TIM1EOF**: TIM1 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM1EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 21:20 **TIM1VC[1:0]**: TIM1 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM1 counter is linked.

00: Virtual channel 0 used to generate the TIM1 event

01: Virtual channel 1 used to generate the TIM1 event

10: Virtual channel 2 used to generate the TIM1 event

11: Virtual channel 3 used to generate the TIM1 event

Note: The write access to these bits is ignored if TIM1EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 19 **TIM0EN**: TIM0 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM0EOF. An interrupt is generated if TIM0IE is set in CSI_IER0.

0: Timer 0 clock cycle counter not active

1: Timer 0 clock cycle counter active

Note: Enabling this bit locks TIM0EOF and TIM0VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 18 **TIM0EOF**: TIM0 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected e as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM0EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 17:16 **TIM0VC[1:0]**: TIM0 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM0 counter is linked.

00: Virtual channel 0 used to generate the TIM0 event

01: Virtual channel 1 used to generate the TIM0 event

10: Virtual channel 2 used to generate the TIM0 event

11: Virtual channel 3 used to generate the TIM0 event

Note: The write access to these bits is ignored if TIM0EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 15 **LB3EN**: Line/byte 3 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB3CFGR. An interrupt is generated if LB3IE is set in CSI_IER0.

0: Line/byte 3 counter stopped

1: Line/byte 3 counter active

Note: Enabling this bit locks LB3VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LB3VC[1:0]**: Line/byte counter 3 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 3 is linked.

00: Virtual channel 0 used to generate the line/byte counter 3 event

01: Virtual channel 1 used to generate the line/byte counter 3 event

10: Virtual channel 2 used to generate the line/byte counter 3 event

11: Virtual channel 3 used to generate the line/byte counter 3 event

Note: The write access to these bits is ignored if LB3EN = 1 in CSI_PRGITR.

Bit 11 **LB2EN**: Line/byte 2 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB2CFGR. An interrupt is generated if LB2IE is set in CSI_IER0.

0: Line/byte 2 counter stopped

1: Line/byte 2 counter active

Note: Enabling this bit locks LB2VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **LB2VC[1:0]**: Line/byte counter 2 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 2 is linked.

00: Virtual channel 0 used to generate the line/byte counter 2 event

01: Virtual channel 1 used to generate the line/byte counter 2 event

10: Virtual channel 2 used to generate the line/byte counter 2 event

11: Virtual channel 3 used to generate the line/byte counter 2 event

Note: The write access to these bits is ignored if LB2EN = 1 in CSI_PRGITR.

Bit 7 **LB1EN**: Line/byte 1 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB1CFGR. An interrupt is generated if LB1IE is set in CSI_IER0.

0: Line/byte 1 counter stopped

1: Line/byte 1 counter active

Note: Enabling this bit locks LB1VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **LB1VC[1:0]**: Line/byte counter 1 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 1 is linked.

00: Virtual channel 0 used to generate the line/byte counter 1 event

01: Virtual channel 1 used to generate the line/byte counter 1 event

10: Virtual channel 2 used to generate the line/byte counter 1 event

11: Virtual channel 3 used to generate the line/byte counter 1 event

Note: The write access to these bits is ignored if LB1EN = 1 in CSI_PRGITR.

Bit 3 **LB0EN**: Line/byte 0 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB0CFGR. An interrupt is generated if LB0IE is set in CSI_IER0.

0: Line/byte 0 counter stopped

1: Line/byte 0 counter active

Note: Enabling this bit locks LB0VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **LB0VC[1:0]**: Line/byte counter 0 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 0 is linked.

00: Virtual channel 0 used to generate the line/byte counter 0 event

01: Virtual channel 1 used to generate the line/byte counter 0 event

10: Virtual channel 2 used to generate the line/byte counter 0 event

11: Virtual channel 3 used to generate the line/byte counter 0 event

Note: The write access to these bits is ignored if LB0EN = 1 in CSI_PRGITR.

43.9.11 CSI-2 Host watchdog register (CSI_WDR)

Address offset: 0x78

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNT[31:0]**: Watchdog counter

These bits contains the maximum number of clock cycles (clk_proc) between an EOF and a SOF for a given channel before triggering an interrupt if WDERRIE = 1 in CSI_IER0.

The value 0x000000 disables the watchdog counter.

43.9.12 CSI-2 Host interrupt enable register 0 (CSI_IER0)

Address offset: 0x80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SYNC ERRIE	WD ERRIE	SPKT ERRIE	ID ERRIE	CECC ERRIE	ECC ERRIE	CRC ERRIE	Res.	Res.	CCFIF OFIE	Res.	Res.	Res.	Res.	SPKTIE
	rw	rw	rw	rw	rw	rw	rw			rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOF3 IE	EOF2 IE	EOF1 IE	EOF0 IE	SOF3 IE	SOF2 IE	SOF1 IE	SOF0 IE	TIM3 IE	TIM2 IE	TIM1 IE	TIM0 IE	LB3 IE	LB2 IE	LB1 IE	LB0 IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SYNCERRIE**: Invalid synchronization error interrupt enable

This bit enables the interrupt generation when an invalid synchronization is detected.
It can be a SOF short packet during a frame, or and EOF short packet during an inter-frame.

0: Interrupt on invalid synchronization error disabled

1: Interrupt on invalid synchronization error enabled

Bit 29 **WDERRIE**: Watchdog error interrupt enable
This bit enables the interrupt generation when the time between two packets for a virtual channel is longer than the programmed watchdog.
0: Interrupt on watchdog error disabled
1: Interrupt on watchdog error enabled

Bit 28 **SPKTERRIE**: Short packet error interrupt enable
This bit enables the interrupt generation based on a reception of a shorter packet than expected.
0: Interrupt on short packet error disabled
1: Interrupt on short packet error enabled

Bit 27 **IDERRIE**: Data type ID error interrupt enable
This bit enables the interrupt generation based on a detection of a reserved data type.
0: Interrupt on reception of reserved data type disabled
1: Interrupt on reception of reserved data type enabled

Bit 26 **CECCERRIE**: Corrected ECC error interrupt enable
This bit enables the interrupt generation based on a corrected ECC error detection.
0: Interrupt on corrected ECC error detection disabled
1: Interrupt on corrected ECC error detection enabled

Bit 25 **ECCERRIE**: ECC error interrupt enable
This bit enables the interrupt generation based on an ECC error detection.
0: Interrupt on ECC error detection disabled
1: Interrupt on ECC error detection enabled

Bit 24 **CRCERRIE**: CRC error interrupt enable
This bit enables the interrupt generation based on CRC error detection on a long packet reception.
0: Interrupt on CRC error detection disabled
1: Interrupt on CRC error detection enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCFIFOFIE**: Clock changer FIFO full interrupt enable
This bit enables the interrupt generation based on clock changer FIFO full detection.
0: Interrupt on clock changer FIFO full detection disabled
1: Interrupt on clock changer FIFO full detection enabled

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **SPKTIE**: Short packet interrupt enable
This bit generates an interrupt which notifies that a generic short packet has been received by the CSI-2 Host.
0: Short packet detection interrupt disabled
1: Short packet detection interrupt enabled

Bit 15 **EOF3IE**: EOF for virtual channel 3 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 3.
0: EOF on virtual channel 3 interrupt disabled
1: EOF on virtual channel 3 interrupt enabled

Bit 14 **EOF2IE**: EOF for virtual channel 2 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 2.
0: EOF on virtual channel 2 interrupt disabled
1: EOF on virtual channel 2 interrupt enabled

Bit 13 **EOF1IE**: EOF for virtual channel 1 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 1.
0: EOF on virtual channel 1 interrupt disabled
1: EOF on virtual channel 1 interrupt enabled

Bit 12 **EOF0IE**: EOF for virtual channel 0 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 0.
0: EOF on virtual channel 0 interrupt disabled
1: EOF on virtual channel 0 interrupt enabled

Bit 11 **SOF3IE**: SOF for virtual channel 3 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 3.
0: SOF on virtual channel 3 interrupt disabled
1: SOF on virtual channel 3 interrupt enabled

Bit 10 **SOF2IE**: SOF for virtual channel 2 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 2.
0: SOF on virtual channel 2 interrupt disabled
1: SOF on virtual channel 2 interrupt enabled

Bit 9 **SOF1IE**: SOF for virtual channel 1 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 1.
0: SOF on virtual channel 1 interrupt disabled.
1: SOF on virtual channel 1 interrupt enabled.

Bit 8 **SOF0IE**: SOF for virtual channel 0 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 0.
0: SOF on virtual channel 0 interrupt disabled
1: SOF on virtual channel 0 interrupt enabled

Bit 7 **TIM3IE**: Timer 3 interrupt enable
This bit enables the interrupt generation from the timer 3 linked to the virtual channel number configured by TIM3VC in CSI_PRGITR. The interrupt raises when the timer 3 value reaches the value configured in CSI_TIM3CFGR.
0: Timer 3 interrupt disabled
1: Timer 3 interrupt enabled

Bit 6 **TIM2IE**: Timer 2 interrupt enable
This bit enables the interrupt generation from the timer 2 linked to the virtual channel number configured by TIM2VC in CSI_PRGITR. The interrupt raises when the timer 2 value reaches the value configured in CSI_TIM2CFGR.
0: Timer 2 interrupt disabled
1: Timer 2 interrupt enabled

Bit 5 **TIM1IE**: Timer 1 interrupt enable

This bit enables the interrupt generation from the timer 1 linked to the virtual channel number configured by TIM1VC in CSI_PRGITR. The interrupt raises when the timer 1 value reaches the value configured in CSI_TIM1CFGR.

0: Timer 1 interrupt disabled

1: Timer 1 interrupt enabled

Bit 4 **TIM0IE**: Timer 0 interrupt enable

This bit enables the interrupt generation from the timer 0 linked to the virtual channel number configured by TIM0VC in CSI_PRGITR. The interrupt raises when the timer 0 value reaches the value configured in CSI_TIM0CFGR.

0: Timer 0 interrupt disabled

1: Timer 0 interrupt enabled

Bit 3 **LB3IE**: Line/byte counter 3 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB3VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB3CFGR.

Bit 2 **LB2IE**: Line/byte counter 2 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB2VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB2CFGR.

Bit 1 **LB1IE**: Line/byte counter 1 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB1VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB1CFGR.

Bit 0 **LB0IE**: Line/byte counter 0 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB0VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB0CFGR.

43.9.13 CSI-2 Host interrupt enable register 1 (CSI_IER1)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ECTRLDL1IE	ESYNCESCDL1IE	EESCDL1IE	ESOTSYNCDL1IE	ESOTDL1IE	Res.	Res.	Res.	ECTRLDL0IE	ESYNCESCDL0IE	EESCDL0IE	ESOTSYNCDL0IE	ESOTDL0IE
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ECTRLDL1IE**: D-PHY_RX lane 1 control error interrupt enable
This bit enables the D-PHY_RX control error interrupt detected on lane 1 consecutive to an incorrect line state sequence.
0: Lane 1 control error interrupt disabled
1: Lane 1 control error interrupt enabled

Bit 11 **ESYNCESCDL1IE**: D-PHY_RX lane 1 low-power data transmission synchronization error interrupt enable
This bit enables the interrupt generation when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.
0: Lane 1 low-power data transmission interrupt error disabled
1: Lane 1 low-power data transmission Interrupt error enabled

Bit 10 **EESCDL1IE**: D-PHY_RX lane 1 escape entry error interrupt enable
This bit enables the interrupt generation when an unrecognized escape entry command is received on lane 1.
0: Lane 1 unrecognized escape entry command interrupt error disabled
1: Lane 1 unrecognized escape entry command interrupt error enabled

Bit 9 **ESOTSYNCDL1IE**: SOT synchronization interrupt error enable on lane 1
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.
0: Lane 1 SOT synchronization interrupt error disabled
1: Lane 1 SOT synchronization interrupt error enabled

Bit 8 **ESOTDL1IE**: SOT error interrupt enable on lane 1
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.
0: Lane 1 SOT interrupt error disabled
1: Lane 1 SOT interrupt error enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ECTRLDL0IE**: D-PHY_RX lane 0 control error interrupt enable
This bit enables the D-PHY_RX control error interrupt detected on lane 0 consecutive to an incorrect line state sequence.
0: Lane 0 control error interrupt disabled
1: Lane 0 control error interrupt enabled

Bit 3 **ESYNCESCDL0IE**: D-PHY_RX lane 0 low power data transmission synchronization error interrupt enable
This bit enables the interrupt generation when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.
0: Lane 0 low-power data transmission interrupt error disabled
1: Lane 0 low-power data transmission Interrupt error enabled

Bit 2 **EESCDL0IE**: D-PHY_RX lane 0 escape entry error interrupt enable
This bit enables the interrupt generation when an unrecognized escape entry command is received on lane 0.
0: Lane 0 unrecognized escape entry command interrupt error disabled
1: Lane 0 unrecognized escape entry command interrupt error enabled

Bit 1 **ESOTSYNCDL0IE**: SOT synchronization interrupt error enable on lane 0
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.
0: Lane 0 SOT synchronization interrupt error disabled
1: Lane 0 SOT synchronization interrupt error enabled

Bit 0 **ESOTDL0IE**: SOT error interrupt enable on lane 0
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.
0: Lane 0 SOT interrupt error disabled
1: Lane 0 SOT interrupt error enabled

43.9.14 CSI-2 Host status register 0 (CSI_SR0)

Address offset: 0x90

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SYNCERRF	WDERRF	SPKTERRF	IDERRF	CECCERRF	ECCERRF	CRCERRF	Res.	Res.	CCFIFOFF	VC3STATEF	VC2STATEF	VC1STATEF	VC0STATEF	SPKTF
	r	r	r	r	r	r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOF3F	EOF2F	EOF1F	EOF0F	SOF3F	SOF2F	SOF1F	SOF0F	TIM3F	TIM2F	TIM1F	TIM0F	LB3F	LB2F	LB1F	LB0F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SYNCERRF**: Invalid synchronization error flag
This bit indicates an invalid synchronization is detected. It can be a SOF short packet during a frame, or and EOF short packet during an inter-frame.

Bit 29 **WDERRF**: Watchdog error flag
This bit indicates that the time between two packets for a virtual channel is longer than the programmed watchdog.

Bit 28 **SPKTERRF**: Short packet error flag
This bit indicates a reception of a shorter packet than expected.

Bit 27 **IDERRF**: Data type ID error flag
This bit indicates a detection of a reserved data type.

Bit 26 **CECCERRF**: Corrected ECC error flag
This bit indicates a corrected ECC error detection took place in the frame.

Bit 25 **ECCERRF**: ECC error flag
This bit indicates that an ECC error has been detected.

Bit 24 **CRCERRF**: CRC error flag
This bit indicates a CRC error detection on a long packet reception.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCFIFOFF**: Clock changer FIFO full flag
This bit indicates a FIFO full detection in the clock changer.

Bit 20 **VC3STATEF**: Virtual channel 3 state flag
This bit indicates the state of the virtual channel 3.
0: Virtual channel inactive
1: Virtual channel active

Bit 19 **VC2STATEF**: Virtual channel 2 state flag
This bit indicates the state of the virtual channel 2.
0: Virtual channel inactive
1: Virtual channel active

Bit 18 **VC1STATEF**: Virtual channel 1 state flag
This bit indicates the state of the virtual channel 1.
0: Virtual channel inactive
1: Virtual channel active

Bit 17 **VC0STATEF**: Virtual channel 0 state flag
This bit indicates the state of the virtual channel 0.
0: Virtual channel inactive
1: Virtual channel active

Bit 16 **SPKTF**: Short packet flag
This bit notifies that a generic short packet has been received by the CSI-2 Host.

Bit 15 **EOF3F**: EOF flag for virtual channel 3
This bit notifies that the CSI-2 Host decodes an EOF on the virtual channel 3.

Bit 14 **EOF2F**: EOF flag for virtual channel 2
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 2.

Bit 13 **EOF1F**:EOF flag for virtual channel 1
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 1.

Bit 12 **EOF0F**: EOF flag for virtual channel 0
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 0.

Bit 11 **SOF3F**: SOF flag for virtual channel 3
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 3.

Bit 10 **SOF2F**: SOF flag for virtual channel 2
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 2.

Bit 9 **SOF1F**: SOF flag for virtual channel 1
This bit notifies that the CSI-2 Host is decoding a sSOF on the virtual channel 1.

Bit 8 **SOF0F**: SOF flag for virtual channel 0
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 0.

Bit 7 **TIM3F**: Timer 3 flag
This bit indicates that the timer 3 linked to the virtual channel configured by TIM3VC in CSI_PRGITR reaches the value programmed in CSI_TIM3CFGR.

Bit 6 **TIM2F**: Timer 2 flag
This bit indicates that the timer 2 linked to the virtual channel configured by TIM2VC in CSI_PRGITR reaches the value programmed in CSI_TIM2CFGR.

Bit 5 **TIM1F**: Timer 1 flag
This bit indicates that the timer 1 linked to the virtual channel configured by TIM1VC in CSI_PRGITR reaches the value programmed in CSI_TIM1CFGR.

Bit 4 **TIM0F**: Timer 0 flag
This bit indicates that the timer 0 linked to the virtual channel configured by TIM0VC in CSI_PRGITR reaches the value programmed in CSI_TIM0CFGR.

Bit 3 **LB3F**: Line/byte counter 3 flag
This bit notifies that the active frame of the virtual channel configured by LB3VC in CSI_PRGITR has reached the line/byte number programmed in th CSI_LB3CFGR.

Bit 2 **LB2F**: Line/byte counter 2 flag
This bit notifies that the active frame of the virtual channel configured by LB2VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB2CFGR.

Bit 1 **LB1F**: Line/byte counter 1 flag
This bit notifies that the active frame of the virtual channel configured by LB1VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB1CFGR.

Bit 0 **LB0F**: Line/byte counter 0 flag
This bit notifies that the active frame of the virtual channel configured by LB0VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB0CFGR.

43.9.15 CSI-2 Host status register 1 (CSI_SR1)

Address offset: 0x94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTCLF	ULPNCLF	ULPNACTF	STOPCLF	Res.	ULPNDL1F	STOPDL1F	SKCALDL1F	SYNCDL1F	ACTDL1F	Res.	ULPNDL0F	STOPDL0F	SKCALDL0F	SYNCDL0F	ACTDL0F
r	r	r	r		r	r	r	r	r		r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ECTRLDL1F	ESYNCESCDL1F	EESCDL1F	ESOTSYNCDL1F	ESOTDL1F	Res.	Res.	Res.	ECTRLDL0F	ESYNCESCDL0F	EESCDL0F	ESOTSYNCDL0F	ESOTDL0F
			r	r	r	r	r				r	r	r	r	r

Bit 31 **ACTCLF**: D-PHY_RX receiver clock active flag
This bit indicates the status of RxClkactiveHs D-PHY_RX signal.

Bit 30 **ULPNCLF**: D-PHY_RX receiver Ultra-Low power state (not) on clock lane.
This bit indicates the status of RxUlpsClkNot D-PHY_RX signal.

Bit 29 **ULPNACTF**: D-PHY_RX receiver ULP state (not) active
This bit indicates the status of UlpsactiveNotC D-PHY_RX signal.

Bit 28 **STOPCLF**: D-PHY_RX receiver in stop state for the clock lane
This bit indicates the status of StopStateC D-PHY_RX signal.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ULPNDL1F**: D-PHY_RX receiver ultra-low-power state (not) active on data lane 1
This bit indicates the status of UlpsActiveNot_D_lane1 D-PHY_RX signal.

Bit 25 **STOPDL1F**: D-PHY_RX receiver data lane 1 in stop state
This bit indicates the status of StopState_D_lane1 D-PHY_RX signal.

Bit 24 **SKCALDL1F**: D-PHY_RX lane 1 high-speed skew calibration
This bit indicates the status of RxSkewCalHS_lane1 D-PHY_RX signal.

Bit 23 **SYNCDL1F**: D-PHY_RX lane 1 receiver synchronization observed
This bit indicates the status of RXSyncHS_lane1 D-PHY_RX signal.

Bit 22 **ACTDL1F**: D-PHY_RX lane 1 high-speed reception active
This bit indicates the status of RxActiveHS_lane1 D-PHY_RX signal.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ULPNDL0F**: D-PHY_RX receiver ultra-low-power state (not) active on data lane 0
This bit indicates the status of UlpsActiveNot_D_lane0 D-PHY_RX signal.

Bit 19 **STOPDL0F**: D-PHY_RX receiver data lane 0 in stop state
This bit indicates the status of StopState_D_lane0 D-PHY_RX signal.

Bit 18 **SKCALDL0F**: D-PHY_RX lane 0 high-speed skew calibration
This bit indicates the status of RxSkewCalHS_lane0 D-PHY_RX signal.

Bit 17 **SYNCDL0F**: D-PHY_RX lane 0 receiver synchronization observed
This bit indicates the status of RXSyncHS_lane0 D-PHY_RX signal.

Bit 16 **ACTDL0F**: D-PHY_RX lane 0 high-speed reception active
This bit indicates the status of RxActiveHS_lane0 D-PHY_RX signal.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ECTRLDL1F**: D-PHY_RX lane 1 control error flag
This bit indicates that the D-PHY_RX detected on lane 1 an incorrect line state sequence.

Bit 11 **ESYNCESCDL1F**: D-PHY_RX lane 1 low-power data transmission synchronization error flag
This bit notifies a synchronization error when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.

Bit 10 **EESCDL1F**: D-PHY_RX lane 1 escape entry error flag
This bit notifies an escape entry error when an unrecognized escape entry command is received on lane 1.

Bit 9 **ESOTSYNCDL1F**: SOT synchronization error flag on lane 1
This bit indicates a SOT synchronization error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.

Bit 8 **ESOTDL1F**: SOT error flag on lane 1
This bit notifies a SOT error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ECTRLDL0F**: D-PHY_RX lane 0 control error flag
This bit indicates that the D-PHY_RX detected on lane 0 an incorrect line state sequence.

Bit 3 **ESYNCESCDL0F**: D-PHY_RX lane 0 low-power data transmission synchronization error flag
This bit notifies a synchronization error when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 0.

Bit 2 **EESCDL0F**: D-PHY_RX lane 0 escape entry error flag
This bit notifies an escape entry error when an unrecognized escape entry command is received on lane 0.

Bit 1 **ESOTSYNCDL0F**: SOT synchronization error flag on lane 0
This bit indicates a SOT synchronization error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.

Bit 0 **ESOTDL0F**: SOT error flag on lane 0
This bit notifies a SOT error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.

43.9.16 CSI-2 Host flag clear register 0 (CSI_FCR0)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CSYNCERRF	CWDERRF	CSPKTERRF	CIDERRF	CCECCERRF	CECCERRF	CCRCERRF	Res.	Res.	CCCFIFOFF	Res.	Res.	Res.	Res.	CSPKTF
	w	w	w	w	w	w	w			w					w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEOF3F	CEOF2F	CEOF1F	CEOF0F	CSOF3F	CSOF2F	CSOF1F	CSOF0F	CTIM3F	CTIM2F	CTIM1F	CTIM0F	CLB3F	CLB2F	CLB1F	CLB0F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CSYNCERRF**: Clear invalid synchronization error flag
Write 1 clears SYNCERRF flag in CSI_SR0.

Bit 29 **CWDERRF**: Clear watchdog error flag
Write 1 clears WDERRF flag in CSI_SR0.

Bit 28 **CSPKTERRF**: Clear short packet error flag
Write 1 clears SPKTERRF flag in CSI_SR0.

Bit 27 **CIDERRF**: Clear data type ID error flag
Write 1 clears CIDERRF flag in CSI_SR0.

Bit 26 **CCECCERRF**: Clear corrected ECC error flag
Write 1 clears CCECCERRF flag in CSI_SR0.

Bit 25 **CECCERRF**: Clear ECC error flag
Write 1 clears CECCERRF flag in CSI_SR0.

Bit 24 **CCRCERRF**: Clear CRC error flag
Write 1 clears CRCERRF flag in CSI_SR0.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCCFIFOFF**: Clear clock changer FIFO full flag
Write 1 clears CCFIFOFF flag in CSI_SR0.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **CSPKTF**: Clear short packet flag
Write 1 clears SPKTF flag in CSI_SR0.

Bit 15 **CEOF3F**: Clear EOF flag for virtual channel 3
Write 1 clears EOF3F flag in CSI_SR0.

Bit 14 **CEOF2F**: Clear EOF flag for virtual channel 2
Write 1 clears EOF2F flag in CSI_SR0.

Bit 13 **CEOF1F**: Clear EOF flag for virtual channel 1
Write 1 clears EOF1F flag in CSI_SR0.

Bit 12 **CEOF0F**: Clear EOF flag for virtual channel 0
Write 1 clears EOF0F flag in CSI_SR0.

Bit 11 **CSOF3F**: Clear SOF flag for virtual channel 3
Write 1 clears SOF3F flag in CSI_SR0.

Bit 10 **CSOF2F**: Clear SOF flag for virtual channel 2
Write 1 clears SOF2F flag in CSI_SR0.

Bit 9 **CSOF1F**: Clear SOF flag for virtual channel 1
Write 1 clears SOF1F flag in CSI_SR0.

Bit 8 **CSOF0F**: Clear SOF flag for virtual channel 0
Write 1 clears SOF0F flag in CSI_SR0.

Bit 7 **CTIM3F**: Clear timer 3 flag
Write 1 clears TIM3F flag in CSI_SR0.

Bit 6 **CTIM2F**: Clear timer 2 flag
Write 1 clears TIM2F flag in CSI_SR0.

Bit 5 **CTIM1F**: Clear timer 1 flag
Write 1 clears TIM1F flag in CSI_SR0.

Bit 4 **CTIM0F**: Clear timer 0 flag
Write 1 clears TIM0F flag in CSI_SR0.

Bit 3 **CLB3F**: Clear line/byte counter 3 flag
Write 1 clears LB3F flag in CSI_SR0.

Bit 2 **CLB2F**: Clear line/byte counter 2 flag
Write 1 clears LB2F flag in CSI_SR0.

Bit 1 **CLB1F**: Clear line/byte counter 1 flag
Write 1 clears LB1F flag in CSI_SR0.

Bit 0 **CLB0F**: Clear line/byte counter 0 flag
Write 1 clears LB0F flag in CSI_SR0.

43.9.17 CSI-2 Host flag clear register 1 (CSI_FCR1)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CECTRLDL1F	CESYNCESCDL1F	CEESCDL1F	CESOTSYNCDL1F	CESOTDL1F	Res.	Res.	Res.	CECTRLDL0F	CESYNCESCDL0F	CEESCDL0F	CESOTSYNCDL0F	CESOTDL0F
			w	w	w	w	w				w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CECTRLDL1F**: Clear D-PHY_RX lane 1 control error flag
Write 1 clears ECTRLDL1F flag in CSI_SR1.

Bit 11 **CESYNCESCDL1F**: Clear D-PHY_RX lane 1 low-power data transmission synchronization error flag
Write 1 clears ESYNCESCDL1F flag in CSI_SR1.

Bit 10 **CEESCDL1F**: Clear D-PHY_RX lane 1 escape entry error flag
Write 1 clears the EESCDL1F flag in the CSI_SR1 register

Bit 9 **CESOTSYNCDL1F**: Clear SOT synchronization error flag on lane 1
Write 1 clears ESOTSYNCDL1F flag in CSI_SR1.

Bit 8 **CESOTDL1F**: Clear SOT error flag on lane 1
Write 1 clears ESOTDL1F flag in CSI_SR1.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CECTRLDL0F**: Clear D-PHY_RX lane 0 control error flag
Write 1 clears ECTRLDL0F in CSI_SR1.

Bit 3 **CESYNCESCDL0F**: Clear D-PHY_RX lane 0 low-power data transmission synchronization error flag
Write 1 clears ESYNCESCDL0F flag in CSI_SR1.

Bit 2 **CEESCDL0F**: Clear D-PHY_RX lane 0 escape entry error flag
Write 1 clears EESCDL0F flag in CSI_SR1.

Bit 1 **CESOTSYNCDL0F**: Clear SOT synchronization error flag on lane 0
Write 1 clears ESOTSYNCDL0F flag in CSI_SR1.

Bit 0 **CESOTDL0F**: Clear SOT error flag on lane 0
Write 1 clears ESOTDL0F flag in CSI_SR1.

43.9.18 CSI-2 Host short packet data field register (CSI_SPDFR)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCHANNEL[1:0]		DATATYPE[5:0]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAFIELD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **VCHANNEL[1:0]**: Virtual channel
This bit indicates which is the virtual channel number decoded in the last generic short packet data field reception. Write to this bit has no effect.

Bits 21:16 **DATATYPE[5:0]**: Data type class
This bit indicates which is the data type class and subclass (according to the MIPI specification for CSI-2) decoded in the last generic short packet data field reception. Write to this bit has no effect.

Bits 15:0 **DATAFIELD[15:0]**: Data field
This bit indicates which is the current data field decoded in the last generic short packet data field reception. Write to this bit has no effect.

43.9.19 CSI-2 Host error register 1 (CSI_ERR1)

Address offset: 0x114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDVCERR[1:0]		IDDTERR[5:0]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CECCVCERR [1:0]		CECCDTERR[5:0]						CRCVCERR [1:0]		CRCDTERR [5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **IDVCERR[1:0]**: Virtual channel having ID error
This bit indicates for which virtual channel an ID error has been detected.
IDERRF is set in CSI_SR0 to highlight the error.

Bits 21:16 **IDDTERR[5:0]**: Data type in error
This bit indicates for which data type an ID error has been detected.
IDERRF is set in CSI_SR0 to highlight the error.

Bits 15:14 **CECCVCERR[1:0]**: Virtual channel having a corrected ECC error
This bit indicates for which virtual channel a corrected ECC error has been detected.
CECCERRF is set in CSI_SR0 to highlight the error.

Bits 13:8 **CECCDTERR[5:0]**: Data type having a corrected ECC error
This bit indicates for which data type a corrected ECC error has been detected.
CECCERRF is set in CSI_SR0 to highlight the error.

Bits 7:6 **CRCVCERR[1:0]**: Virtual channel having a CRC error
This bit indicates for which virtual channel a CRC error has been detected.
CRCERRF is set in CSI_SR0 to highlight the error.

Bits 5:0 **CRCDTERR[5:0]**: Data type having a CRC error
This bit indicates for which data type a CRC error has been detected.
CRCERRF is set in CSI_SR0 to highlight the error.

43.9.20 CSI-2 Host error register 2 (CSI_ERR2)

Address offset: 0x118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCVCERR [1:0]		WDVCERR [1:0]	
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPKTVCERR [1:0]		SPKTDTERR [5:0]					
								r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **SYNCVCERR[1:0]**: Virtual channel having synchronization error

This bit indicates for which virtual channel a synchronization error has been detected. SYNCERRF is set in CSI_SR0 to highlight the error.

Bits 17:16 **WDVCERR[1:0]**: Virtual channel having a watchdog error

This bit indicates for which virtual channel a watchdog error has been detected. WDERRF is set in CSI_SR0 to highlight the error.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **SPKTVCERR[1:0]**: Virtual channel having a short packet error

This bit indicates for which virtual channel a short packet error has been detected. SPKTERRF is set in CSI_SR0 to highlight the error.

Bits 5:0 **SPKTDTERR[5:0]**: Data type having a short packet error

This bit indicates for which data type a shorter packet than expected error has been detected. SPKTERRF is set in CSI_SR0 to highlight the error.

43.9.21 CSI-2 Host hardware configuration register (CSI_HHWCFGR)

Address offset: 0x0FF0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TECHNO[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TECHNO[3:0]**: Technology
This field returns the technology used.
0x0: Technology information is not provided.
0x1: Reserved for 40nm PHY
0x2: 16FF Synopsys PHY

43.9.22 CSI-2 Host version register (CSI_HVERR)

Address offset: 0x0FF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
This field returns the major revision of the CSI-2 Host.

Bits 3:0 **MINREV[3:0]**: Major revision
This field returns the minor revision of the CSI-2 Host.

43.9.23 CSI-2 Host identification register (CSI_HIPIDR)

Address offset: 0x0FF8

Reset value: 0x0016 2011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code
This field returns the identification code of the CSI-2 Host.

43.9.24 CSI-2 Host size identification register (CSI_HSIDR)

Address offset: 0x0FFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size and ID

This field returns the size and the ID of the CSI-2 Host.

43.10 CSI PHY registers

43.10.1 CSI PHY reset control register (CSI_PRCR)

Address offset: 0x1000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PEN**:

When set to 0, this bit places the digital section of the D-PHY in the reset state.

0: PHY is disabled (in reset state).

1: PHY is enabled (out of reset state).

Bit 0 Reserved, must be kept at reset value.

43.10.2 CSI PHY mode control register (CSI_PMCR)

Address offset: 0x1004

Reset value: 0x0001 1003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TUEXDL0
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TUESDL0	Res.	Res.	Res.	RTDL0	Res.	Res.	Res.	DTDL	Res.	FTXSMDL0	FRXMDL1	FRXMDL0
			rw				rw				rw		rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TUEXDL0**: Tx ULP exit sequence data lane 0
This bit initiates the transmission of the ULP exit sequence.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TUESDL0**: Tx ULP escape-mode data lane 0

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **RTDL0**: Turn-around request data lane 0
This bit is used to indicate that the protocol layer needs to turn the lane around, allowing from the other side to start transmitting the data.
0: No request
1: Request for turn-around for DL0

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **DTDL**: Disable turn-around data lane 0
This bit is used to prevent the bi-directional lane from processing a turn-around request in the lane interconnect
0: Disabled
1: Enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTXSMDL0**: Force to Tx Stop mode the data lane 0
0: Disabled
1: Enabled

Bit 1 **FRXMDL1**: Force to Rx mode the data lane 1
Module 0 into receive mode/wait for stop state.
0: Disabled
1: Enabled

Bit 0 **FRXMDL0**: Force to Rx mode the data lane 0
Module 0 into receive mode/wait for stop state.
0: Disabled
1: Enabled

43.10.3 CSI PHY frequency control register (CSI_PFCR)

Address offset: 0x1008

Reset value: 0x0000 0017

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLD
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HSFR[6:0]							Res.	Res.	CCFR[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DLD**: Data lane direction of lane 0
0: Rx
1: Tx

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **HSFR[6:0]**: PHY high-speed frequency range selection
Must be set when the CSI PHY is in power-down, and must not change during operation.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **CCFR[5:0]**: Configuration clock frequency range selection
Must be set at value = round (Fcfg_clk(MHz) - 17) * 4.

43.10.4 CSI PHY test control register 0 (CSI_PTCR0)

Address offset: 0x1010

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSEN	TCKEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **TRSEN**: Test-interface reset enable for the TDI bus into the PHY

Bit 0 **TCKEN**: Test-interface clock enable for the TDI bus into the PHY

43.10.5 CSI PHY test control register 1 (CSI_PTCR1)

Address offset: 0x1014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TWM
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDI[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TWM**: Test-interface write mode selector
0: Data write operation is set on the rising edge of the testclk signal.
1: Address write operation is set on the falling edge of the testclk signal.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TDI[7:0]**: Test-interface data in

43.10.6 CSI PHY test status register (CSI_PTSR)

Address offset: 0x1018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDO[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDO[7:0]**: CSI PHY test interface data output bus for read-back and internal probing functionalities

43.10.7 CSI-2 Host hardware configuration register (CSI_HWCFGR)

Address offset: 0x1FF0

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LANES[3:0]				TECHNO[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **LANES[3:0]**: Amount of CSI data lanes (excluding the clock lane)
0x0: Lane information not provided
0x1: One CSI data lane implemented
0x2: Two CSI data lanes implemented
0x4: Four CSI data lanes implemented

Bits 3:0 **TECHNO[3:0]**: Technology
This field returns the technology used.
0x0: Technology information not provided
0x1: Reserved for 40nm PHY
0x2: 16FF Synopsys PHY.

43.10.8 CSI-2 Host version register (CSI_VERR)

Address offset: 0x1FF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
This field returns the major revision of the CSI-2 Host.

Bits 3:0 **MINREV[3:0]**: Major revision
This field returns the minor revision of the CSI-2 Host.

43.10.9 CSI-2 Host identification register (CSI_IPIDR)

Address offset: 0x1FF8

Reset value: 0x0016 2011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code
This field returns the identification code of the CSI-2 Host.

43.10.10 CSI-2 Host size identification register (CSI_SIDR)

Address offset: 0x1FFC

Reset value: 0xA3C5 DD08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size and ID

This field returns the size and the ID of the CSI-2 Host.

43.10.11 CSI-2 Host and PHY register map

Table 418. CSI-2 Host and PHY register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CSI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC3STOP	VC3START	Res.	Res.	VC2STOP	VC2START	Res.	Res.	VC1STOP	VC1START	Res.	Res.	VC0STOP	VC0START	Res.	CSIEN.
	Reset value																	0	0			0	0			0	0			0	0		0
0x04	CSI_PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1EN	DL0EN	CLEN	PWRDOWN
	Reset value																													0	0	0	0
0x08-0x0C	Reserved	Reserved																															
0x10 + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR1	Res.	Res.	Res.	DT0FT[4:0]					Res.	Res.	DT0[5:0]						Res.	Res.	Res.	CDTFT[4:0]					DT6EN	DT5EN	DT4EN	DT3EN	DT2EN	DT1EN	DT0EN	ALLDT
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR2	Res.	Res.	Res.	DT2FT[4:0]					Res.	Res.	DT2[5:0]						Res.	Res.	Res.	DT1FT[4:0]					Res.	Res.	DT1[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x18+ 0x10*x (x=0 t0 3)	CSI_ VCxCFGR3	Res.	Res.	Res.	DT4FT[4:0]					Res.	Res.	DT4[5:0]						Res.	Res.	Res.	DT3FT[4:0]					Res.	Res.	DT3[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x1C + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR4	Res.	Res.	Res.	DT6FT[4:0]					Res.	Res.	DT6[5:0]						Res.	Res.	Res.	DT5FT[4:0]					Res.	Res.	DT5[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x50 + 0x04*x (x=0 t0 3)	CSI_LBxCFGR	LINECNT[15:0]																BYTECNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 418. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60 + 0x04*x (x=0 t0 3)	CSI_ TIMxCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CSI_LMCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1MAP[2:0]			Res.		DL0MAP[2:0]		Res.	Res.	Res.	Res.	Res.	LANENB[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	1	0	0	0	0	1					0	0	1	0								
0x74	CSI_PRGITR	TIM3EN	TIM3EOF	TIM3VC[1:0]		TIM2EN	TIM2EOF	TIM2VC[1:0]		TIM1EN	TIM1EOF	TIM1VC[1:0]		TIM0EN	TIM0EOF	TIM0VC[1:0]		LB3EN	Res.	LB3VC[1:0]		LB2EN	Res.	LB2VC[1:0]		LB1EN	Res.	LB1VC[1:0]		LB0EN	Res.	LB0VC[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0
0x78	CSI_WDR	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	Reserved	Reserved																															
0x80	CSI_IER0	Res.	SYNCERRIE	WDERRIE	SPKTERRIE	IDERRIE	CECCERRIE	ECCERRIE	CRCERRIE	Res.	Res.	CCFIFOFIE	Res.	Res.	Res.	Res.	SPKTIE	EOF4IE	EOF3IE	EOF2IE	EOF1IE	SOF3IE	SOF2IE	SOF1IE	SOF0IE	TIM3IE	TIM2IE	TIM1IE	TIM0IE	LB3IE	LB2IE	LB1IE	LB0IE
	Reset value		0	0	0	0	0	0	0			0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CSI_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECTRLDL1IE	ESYNCESCDL1IE	EESCDL1IE	ESOTSYNCDL1IE	ESOTDL1IE	Res.	Res.	Res.	ECTRLDL0IE	ESYNCESCDL0IE	EESCDL0IE	ESOTSYNCDL0IE	ESOTDL0IE
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x88-0x8C	Reserved	Reserved																															
0x90	CSI_SR0	Res.	SYNCERRF	WDERRF	SPKTERRF	IDERRF	CECCERRF	ECCERRF	CRCERRF	Res.	Res.	CCFIFOFF	VC3STATEF	VC2STATEF	VC1STATEF	VC0STATEF	SPKTF	EOF3F	EOF2F	EOF1F	EOF0F	SOF3F	SOF2F	SOF1F	SOF0F	TIM3F	TIM2F	TIM1F	TIM0F	LB3F	LB2F	LB1F	LB0F
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	CSI_SR1	ACTCLF	ULPNCLF	ULPNACTF	STOPCLF	Res.	ULPNDL1F	STOPDL1F	SKCALDL1F	SYNCDL1F	ACTDL1F	Res.	ULPNDL0F	STOPDL0F	SKCALDL0F	SYNCDL0F	ACTDL0F	Res.	Res.	Res.	ECTRLDL1F	ESYNCESCDL1F	EESCDL1F	ESOTSYNCDL1F	ESOTDL1F	Res.	Res.	Res.	ECTRLDL0F	ESYNCESCDL0F	EESCDL0F	ESOTSYNCDL0F	ESOTDL0F
	Reset value	0	0	0	0		0	0	0	0	0		0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x98-0x9C	Reserved	Reserved																															
0x100	CSI_FCR0	Res.	CSYNCERRF	CWDERRF	CSPKTERRF	CIDERRF	CCECCRRF	CECCERRF	CCRCERRF	Res.	Res.	CCCFIFOFF	Res.	Res.	Res.	Res.	CSPKTF	CEOF3F	CEOF2F	CEOF1F	CEOF0F	CSOF3F	CSOF2F	CSOF1F	CSOF0F	CTIM3F	CTIM2F	CTIM1F	CTIM0F	CLB3F	CLB2F	CLB1F	CLB0F
	Reset value		0	0	0	0	0	0	0			0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 418. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x104	CSI_FCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CECTRLDL1F	CESYNCESCDLF	CEESCDL1F	CESOTSYNCDL1F	CESOTDL1F	Res.	Res.	Res.	CECTRLDL0F	CESYNCESCDL0F	CEESCDL0F	CESOTSYNCDL0F	CESOTDL0F
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x108-0x10C	Reserved	Reserved																															
0x110	CSI_SPDFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCHANNEL[1:0]		DATATYPE[5:0]						DATAFIELD[15:0]															
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	CSI_ERR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDVCERR[1:0]		IDDTERR[5:0]						CECCVCERR[1:0]		CECCDTERR[5:0]						CRCVCERR[1:0]		CRCDTERR[5:0]					
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118	CSI_ERR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCVCERR[1:0]		WDVCERR[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPKTVCERR[1:0]		SPKTDTERR[5:0]					
	Reset value													0	0	0	0									0	0	0	0	0	0	0	0
0x11C-0xFEC	Reserved	Reserved																															
0xFF0	CSI_HHWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TECHNO[3:0]			
	Reset value																													0	0	0	0
0xFF4	CSI_HVERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	0
0xFF8	CSI_HIPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
0xFFC	CSI_HSIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0
0x1000	CSI_PRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		PEN	Res.
	Reset value																															0	
0x1004	CSI_PMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TUEXDL0	Res.	Res.	Res.	TUESDL0	Res.	Res.	Res.	RTDL0	Res.	Res.	Res.	DTDL	Res.	FTXSMDL0	FRXMDL1	FRXMDL0
	Reset value																1				1				0				0		0	1	1

Table 418. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1008	CSI_PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLD	Res.	HSFR[6:0]							Res.	Res.	CCFR[5:0]					
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
0x100C	Reserved	Reserved																															
0x1010	CSI_PTCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSEN	TCKEN
	Reset value																															0	1
0x1014	CSI_PTCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDI[7:0]							
	Reset value																0									0	0	0	0	0	0	0	0
0x1018	CSI_PTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDO[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x101C-0x1FEC	Reserved	Reserved																															
0x1FF0	CSI_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LANES[3:0]				TECHNO[3:0]			
	Reset value																									0	0	1	0	0	0	1	0
0x1FF4	CSI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x1FF8	CSI_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
0x1FFC	CSI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	1	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

44 Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see *Section 44.3.2: PSSI pins and internal signals*).

44.1 PSSI introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

44.2 PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) and Ready (PSSI_RDY) alternate function

  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.

44.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

Figure 380 shows the PSSI block diagram.

44.3.1 PSSI block diagram

Figure 380. PSSI block diagram

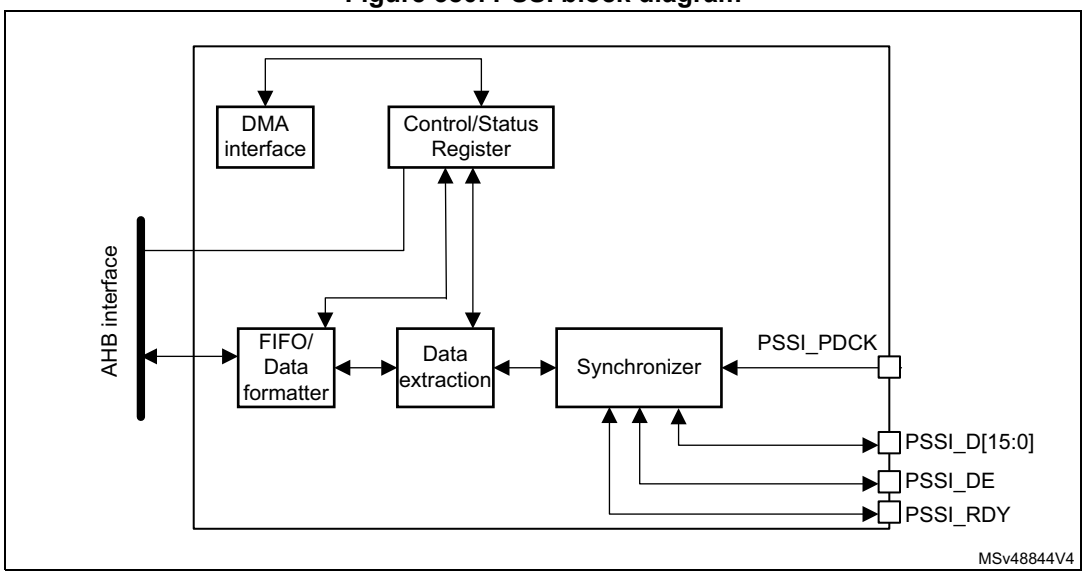
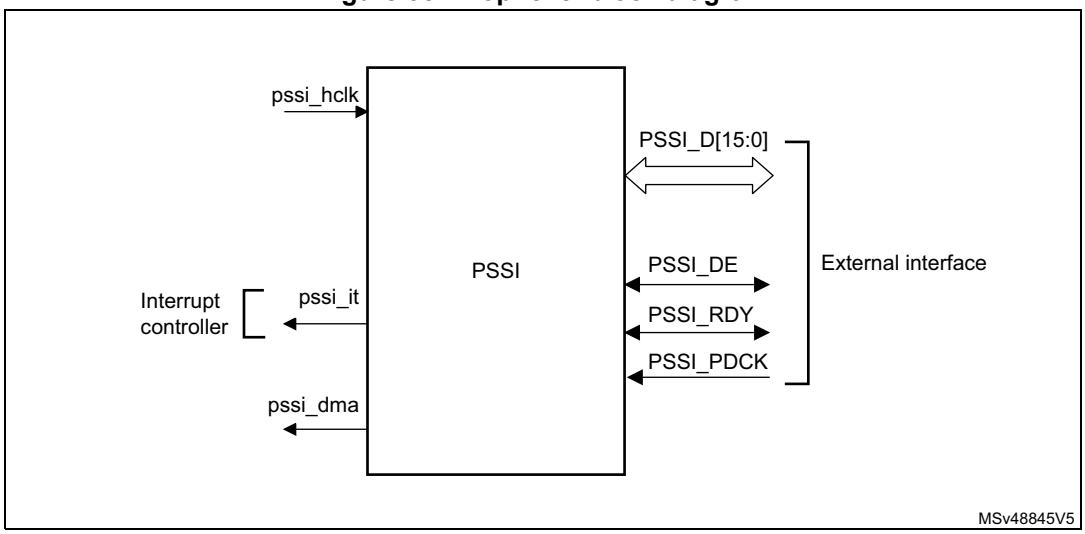


Figure 381. Top-level block diagram



44.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. *Table 419* shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI_CR) and the PSSI ENABLE bit (bit 14 of PSSI_CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.

Table 419. PSSI input/output pins

PSSI signal name	DCMI signal it is shared with	Signal type	Description
PSSI_PDCK	DCMI_PIXCK	Input	Parallel data clock input
PSSI_D[15:0]	DCMI_D[13:0]	Input/output	Data output when transmitting, data input when receiving
PSSI_DE	DCMI_HSYNC	Input/output	Data enable signal
PSSI_RDY	DCMI_VSYNC	Input/output	Ready signal

Table 420 shows the PSSI internal input/output signals.

Table 420. PSSI internal input/output signals

Internal signal name	Signal type	Description
pssi_it	Output	Interrupt
pssi_dma	Output	DMA request
pssi_hclk	Input	AHB clock

44.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

44.3.4 PSSI data management

Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.

Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:

- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly perform transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. *Table 421* illustrates the positioning of the data bytes in two 32-bit words.

Table 421. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. *Table 422* illustrates the positioning of the data in two 32-bit words.

Table 422. Positioning of captured data bytes in 32-bit words (16-bit width)

Byte address	31:16	15:0
0	D_{n+1}[15:0]	D_n[15:0]
4	D_{n+3}[15:0]	D_{n+2}[15:0]

FIFO data buffer and error conditions

An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

44.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

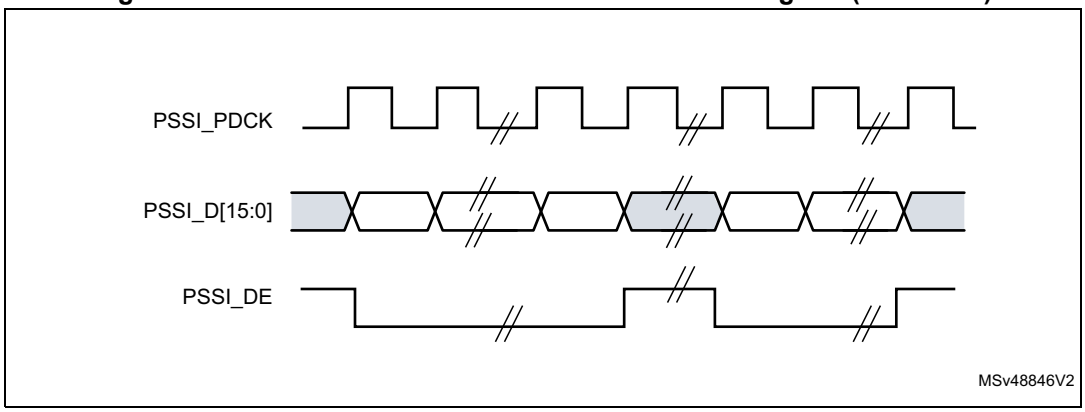
The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is et to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

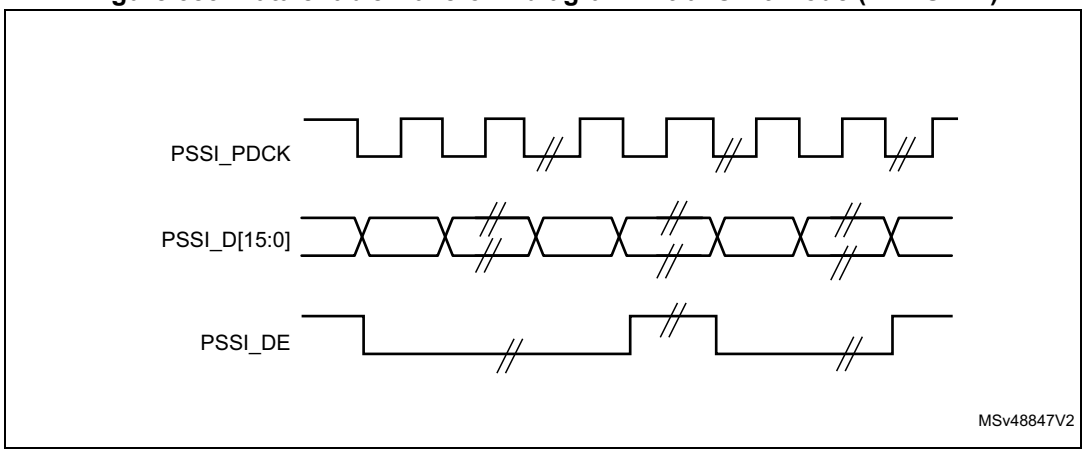
If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.

Figure 382. Data enable in receive mode waveform diagram (CKPOL=0)



If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

Figure 383. Data enable waveform diagram in transmit mode (CKPOL=1)



Ready (PSSI_RDY) alternate function output

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

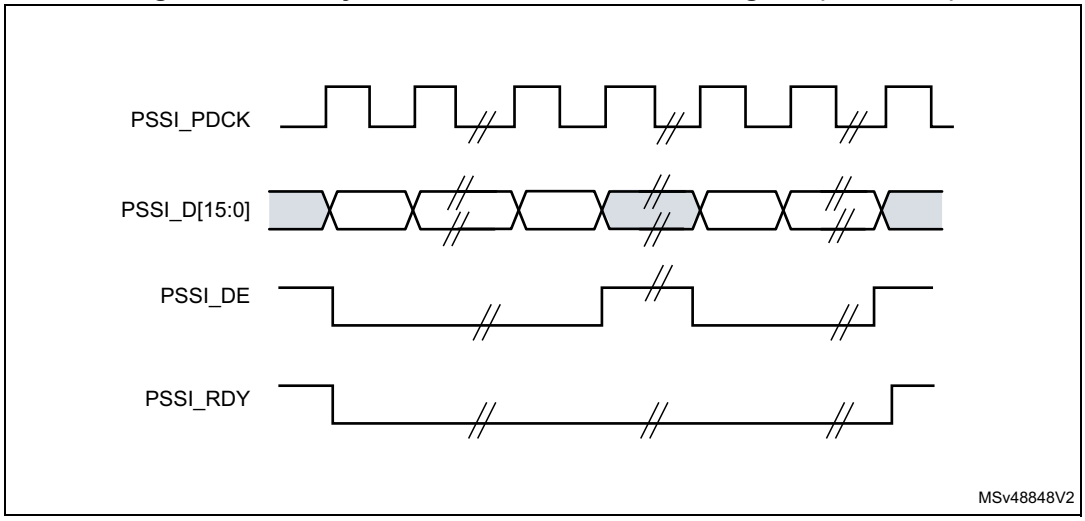
This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples

the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Figure 384. Ready in receive mode waveform diagram (CKPOL=0)



If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Bidirectional PSSI_DE/PSSI_RDY signal

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices know that the data is not being transferred in the current cycle.

Figure 385. Bidirectional PSSI_DE/PSSI_RDY waveform

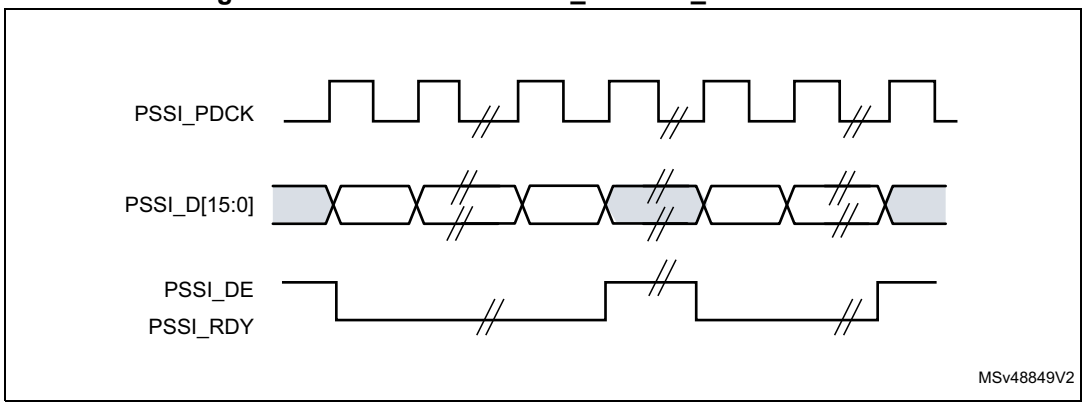
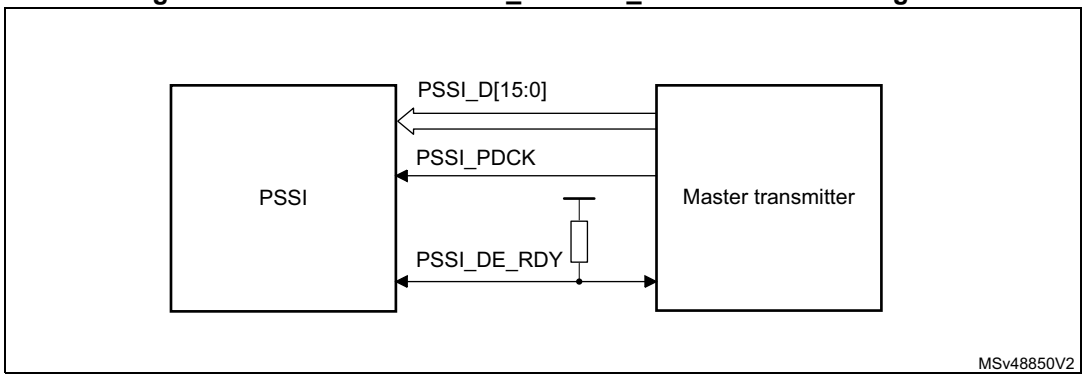


Figure 386. Bidirectional PSSI_DE/PSSI_RDY connection diagram



44.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to *Table 423* for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

Table 423. PSSI interrupt requests

Interrupt acronym	Shared with DCMI	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from low-power mode
IT_OVR	IT_OVR	indicates overrun in receive mode or underrun in transmit mode	OVR_RIS	OVR_IE	OVR_ISC	NA

44.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

44.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUTEN	DMAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG[2:0]			Res.	Res.
rw	rw										rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw		rw		rw	rw					

Bit 31 **OUTEN:** Data direction selection bit

0: Receive mode: data is input synchronously with PSSI_PDCK

1: Transmit mode: data is output synchronously with PSSI_PDCK

Bit 30 **DMAEN:** DMA enable bit

0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.

1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 **DERDYCFG[2:0]:** Data enable and ready configuration

000: PSSI_DE and PSSI_RDY both disabled

001: Only PSSI_RDY enabled

010: Only PSSI_DE enabled

011: Both PSSI_RDY and PSSI_DE alternate functions enabled

100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 2425*)

101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin

110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin

111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 2425*)

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** PSSI enable

0: PSSI disabled

1: PSSI enabled

The contents of the FIFO are flushed when ENABLE is cleared to 0.

Note: When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.

The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.

The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every parallel data clock

01: Reserved, must not be selected

10: Reserved, must not be selected

11: The interface captures 16-bit data on every parallel data clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RDYPOL:** Ready (PSSI_RDY) polarity

This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.

0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)

1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DEPOL:** Data enable (PSSI_DE) polarity

This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.

0: PSSI_DE active low (0 indicates that data is valid)

1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 **CKPOL:** Parallel data clock polarity

This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.

0: Falling edge active for inputs or rising edge active for outputs

1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.

44.5.2 PSSI status register (PSSI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
												r	r		

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RTT1B:** FIFO is ready to transfer one byte

1: FIFO is ready for a one byte transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.

0: FIFO is not ready for a 1-byte transfer

Bit 2 **RTT4B:** FIFO is ready to transfer four bytes

1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.

0: FIFO is not ready for a four-byte transfer

Bits 1:0 Reserved, must be kept at reset value.

44.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_RIS:** Data buffer overrun/underrun raw interrupt status

0: No overrun/underrun occurred

1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode. This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

Bit 0 Reserved, must be kept at reset value.

44.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_IE:** Data buffer overrun/underrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if either an overrun or an underrun error occurred.

Bit 0 Reserved, must be kept at reset value.

44.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_MIS:** Data buffer overrun/underrun masked interrupt status
This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.
0: No interrupt is generated when an overrun/underrun error occurs
1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

Bit 0 Reserved, must be kept at reset value.

44.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_ISC	Res.
														w	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_ISC:** Data buffer overrun/underrun interrupt status clear
Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.

44.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

In receive mode (OUTEN = 0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN = 1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

44.5.8 PSSI hardware configuration register (PSSI_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_8[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **FIFO_8[3:0]:** FIFO size

44.5.9 PSSI version register (PSSI_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** IP version, major revision information

Bits 3:0 **MINREV[3:0]:** IP version, minor revision information

44.5.10 PSSI ID register (PSSI_IPIDR)

Address offset: 0x3F8

Reset value: 0x0016 1011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IP_ID[31:0]**: 00161011 for PSSI

44.5.11 PSSI size identification register (PSSI_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification code

This bitfield returns the size identification code of the AES peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1 Kbyte address decoding)

44.5.12 PSSI register map

Table 424. PSSI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PSSI_CR	OUTEN	DMAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG			Res.	Res.	Res.	ENABLE[2:0]			EDM		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1										0	0	0				0			0	0		0		0	0					
0x04	PSSI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
	Reset value																													0	0		
0x08	PSSI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
	Reset value																															0	

Table 424. PSSI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	PSSI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
	Reset value																															0	
0x10	PSSI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
	Reset value																															0	
0x14	PSSI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_ISC	Res.
	Reset value																															0	
0x18-0x24	Reserved	Res.																															
0x28	PSSI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C-0x3EC	Reserved	Res.																															
0x3F0	PSSI_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_8[3:0]			
	Reset value																													0	0	0	1
0x3F4	PSSI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	1	0	0	1	0
0x3F8	PSSI_IPIDR	IP_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	PSSI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

45 Display subsystem

45.1 Introduction

The display subsystem is targeted to drive up to 1080p60 display panels, through a parallel interface. It supports some on-the-fly compositions and optimizes the use of the system bandwidth: up to three layers, color conversion, blend, mirror, and a final YUV conversion.

The display subsystem can display a secure layer with data that cannot be read by non-secure application, and with the display guaranteed stable.

45.2 Display subsystem main features

The display subsystem is built around LTDC:

- LTDC: handles display composition
  - Composition: three layers
  - Input pixel format: flexible format, including YUV420 full-planar on layer L1
  - Secure layer: protected access to buffer and configuration registers.
  - Mirror: horizontal and vertical
  - Miscellaneous: color lookup-table, color keying, Gamma
  - 1080p60 max performance
- LTDC parallel interface (integrated inside the LTDC):
  - Standard: 24 bpp + Hs, Vs, De (DataEnable) synchronization signals with ReTime
  - Output rate: 150 Mpixel/s
  - Resolution: 1920x1080 at 60 fps max, with HDMI blankings
  - Pixel formats: RGB888, 666, 565, YUV422-16 bits/BT601/709

45.3 Display subsystem implementation

The implementation is described in *Section 46: LCD-TFT display controller (LTDC)*.

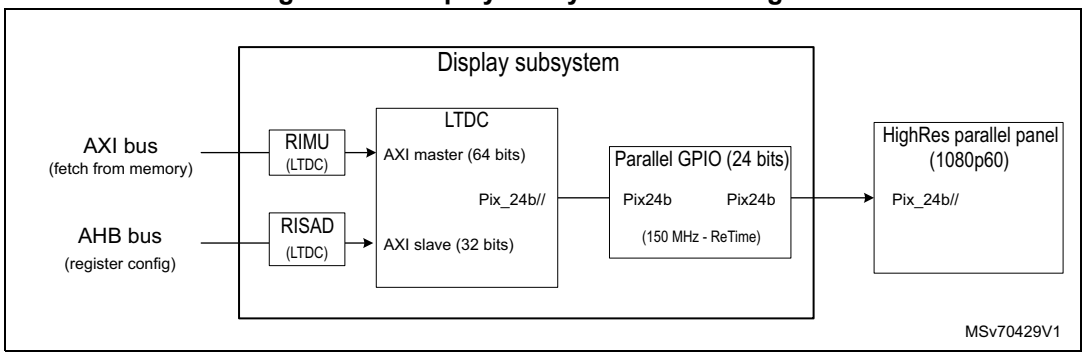
45.4 Display subsystem functional description

45.4.1 Display subsystem block diagram

Figure 387 shows the functional blocks in the display subsystem, with their instance/design names:

- LTDC: composition engine, includes a parallel interface
- Parallel GPIO: Parallel I/O-ring of the device with up to 24 bpp + H/V synchronization
- RIMU/RISAD: RIF security logic, configured for the specific display needs

Figure 387. Display subsystem block diagram

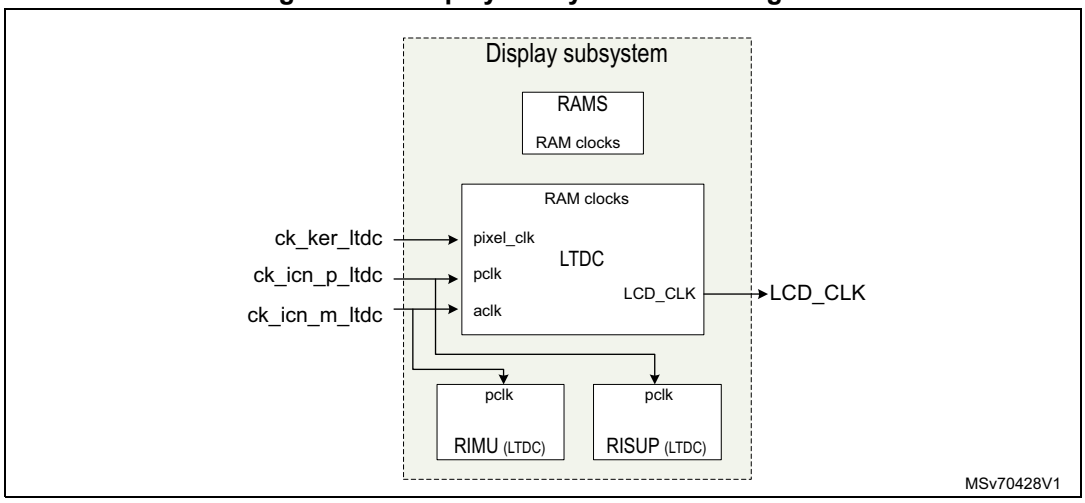


45.4.2 Display subsystem pins and external signal interface

The details of the pins and external signal interface of the display subsystem can be found in the respective functional block sections in this reference manual.

45.4.3 Display subsystem clocks

Figure 388. Display subsystem clock diagram



The display subsystem is implemented with the following clock max frequencies:

- LTDC pixel clock (pixel_clk): 150 MHz
- LTDC parallel interface (lcd_clk): 150 MHz (I/O ReTime mandatory)
- Register clocks (ck_icn_p_*): 150 MHz (all APB/AHB register files)
- AXI-Master clock (ck_icn_m_ltdc): 400 MHz

45.4.4 Display subsystem security

This section is an application of RIF security architecture (see *Section 5.4: Resource isolation framework (RIF)*). The security of the display subsystem is based on the protection of its register map and its master access to the system interconnect, based on default RISUP and RIMU respectively.

The RISUP differentiates the access right of accesses performed toward the following RIF protected peripheral ID (see *Table 38: Indexes for non-RIF-aware peripherals (RISUP)*):

- LTDC common: LTDC common registers, about panel info, synchronization, interface
- LTDC_L1L2 (layer 1 and 2): for the window of any two default applications
- LTDC_L3: LTDC layer 3, for the window of a potentially secure application, or any default application if there is no secure layer

The RIMU (RIMU_AXI_LTDC) differentiates the bus transactions emitted by the following AXI masters:

- RIMU_L1L2: read access only for layer 1 and 2, always non-protected
- RIMU_L3: read access only for layer 3, potentially protected

Security configuration of devices with never-secure display content

The security setup of devices that are never required to display any secure content is detailed below (the display subsystem works by default in a non-protected mode):

- The display subsystem configuration space is set as non-protected.

  The default operating system (OS) software has a full access and can configure/handle all peripherals of the display subsystem.
- The layer 3 (which has a hardware secure capability) can be reused as a default layer, like layer 1 and 2, and can display the content of any application.

Note: Layer 3 is not fully featured, and lacks the resize and multi-planar capabilities of layers 1/2.

- The LTDC AXI-master emits by default non-protected accesses. It inherits NP and NS modes from its associated RIF peripheral ID.

  The AXI CID = DISP can be associated to the three AXI clients, via the RIMU, to tag that the AXI transaction contains pixels.

Security configuration of devices with a temporary secure content

For devices that temporarily display a secure content, it is expected that the secure content is displayed during short periods.

The working principle of the display subsystem is to segregate configuration spaces:

- The secure layer 3 is always protected.
- Never-secure layers 1 and 2 are always non-protected.
- The intermediate common features (such as panel data, synchronization) are usually non-protected and handled by the OS (when no secure content is displayed), but temporarily protected (with no OS access) during the short period when a secure content is displayed via layer 3. Removing the access of the non-protected OS guarantees that a malware does not impact the stability of the displayed protected content.
- A few minor items get specifically protected or open, as detailed below.

The configuration of the display subsystem for a secure content is the following:

- Layer 3 is set permanently protected:
  - RIF LTDC_L3 RISUP peripheral ID must be set as protected (S and/or P flags).
  - RIF LTDC_L3 RIMU AXI-Master emits protected transactions (tagged as S a/o P), which allow it to retrieve the protected layer 3 pixels from their protected memory

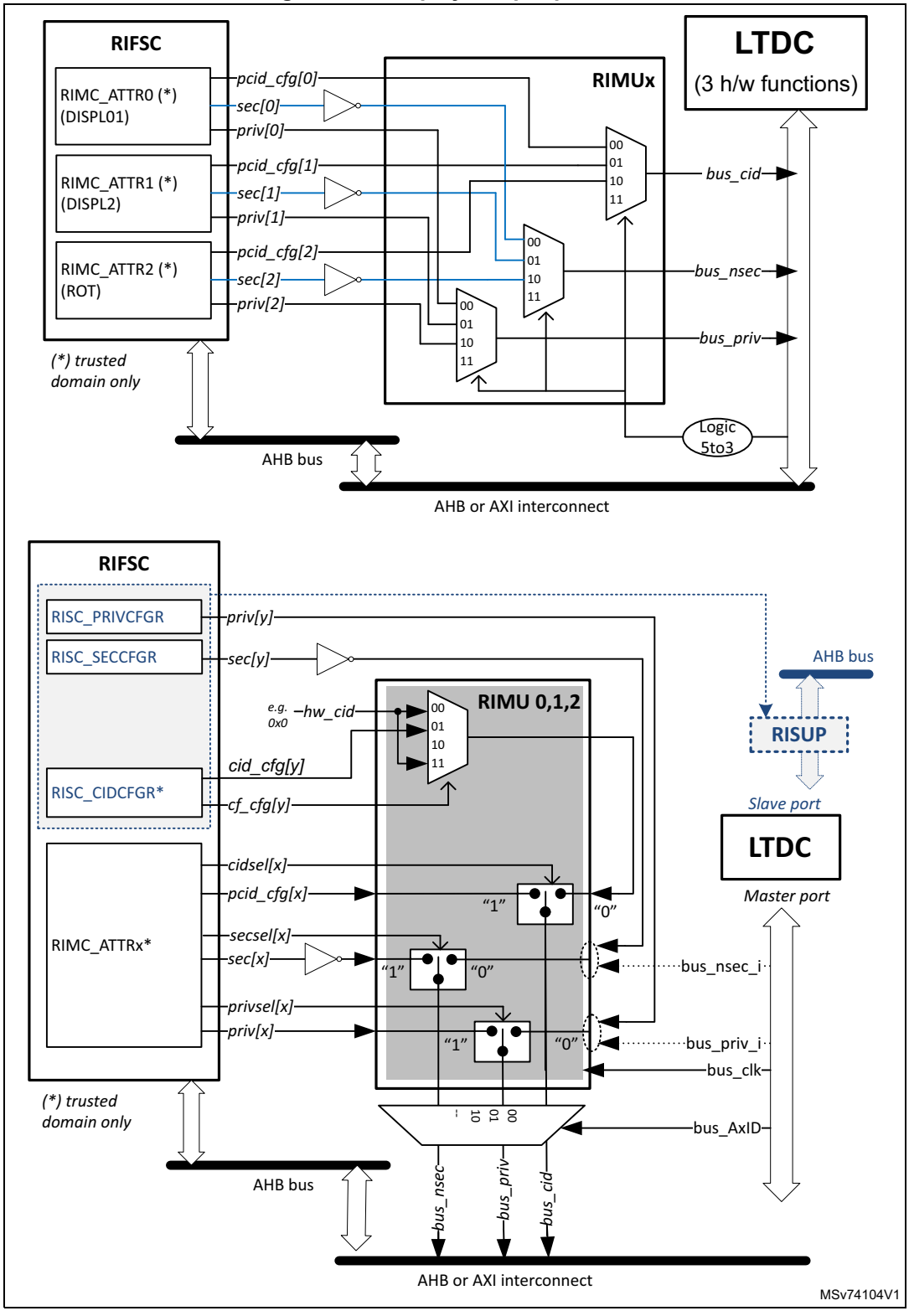
buffer. The RIMU must be updated by a trusted domain application using RIFSC_RIMC_ATTR12.

The recommendation is to use a secure, privileged (optional), and CID = 0x4 setup that is configured in RIFSC_RISC_PER120_CIDCFGR (and optionally RIFSC_RISC_PER120_SEMCR). With this secure setup, the application must make sure that CID4 has access to the secure pixels area.

  - Only a secure software is able to configure the layer 3 parameters (such as width, height, buffer address), so that the characteristics of the layer 3 are guaranteed.
  - The interrupt bank B is associated automatically with the layer 3, and gets permanently protected and devoted to the secure software handling layer 3. The secure software must thus handle and be driven by that interrupt bank B.
  - The LTDC features that must allow data-evasion get also protected (automatically, as associated with the layer 3 rights), with an access allowed only to the secure software. These features are the CRC (which allows the displayed frames to be signed mainly for test purposes), the blend-order (which is used to hide the protected content below some malware layer faking the protected content).
- Layers 1 and 2 are set permanently as non-protected:
  - RIF LTDC_L1L2 RISUP peripheral ID is set as not protected.
  - RIF LTDC_L1L2 RIMU AXI-master keeps its initial setup in RIFSC_RIMC_ATTR11, and emits only non-protected transactions which is enough to retrieve pixels from any application buffer.
  - The default OS drivers handle the layer 1 and 2 parameters (such as width, height, buffer address).
  - The interrupt bank A is associated automatically with layer 1 and 2, and keeps an non-protected open access. The OS must then handle and be driven by that Interrupt bank A to globally drive the display.
- Common registers must be set at boot as non-protected, temporarily:
  - The non-layer peripheral index RIF LTDC common is concerned.
  - This allows the default OS to boot the display subsystem, and to handle it.
- Common registers must be set as protected when displaying a protected content:
  - RIF LTDC common must be set protected before starting to display a protected content, and be unprotected after the protected content has finished to be updated. The LTDC common configuration space keeps a permanent read-access for non-secure or unprivileged software.
  - The protection restricts the write accesses only: it removes the access of the non-protected OS, and so guarantees that a malware does not impact the stability of the displayed protected content. The LTDC common configuration space keeps a permanent read-access for non-secure or unprivileged software (PRDENy bits in RISAB_CIDxRDCFGR can be used).
  - The protection does not restrict the read accesses: the OS is usually unaware of the status of the protected layer 3 that is handled by a different, secure, software. The read access capability for the OS allow the display drivers to remain. somehow stable, although the write access of the display drivers is not performed.

The above recommended settings for the device with temporary secure layer are shown in the figure below.

Figure 389. Display RIF peripheral ID



The table below lists vertically all the functional blocks of the display subsystem. It also lists horizontally the associated RIF peripheral ID.

Note: Table color legend for recommended settings:

- green for a permanently non-protected setting

- blue for a temporarily protected versus non-protected

- purple for a permanently protected state

Table 425. Display subsystem RIF peripheral ID

All in display subsystem					RISUP-slave		
					80	119	120
Blocks	Address range	(LTDC) address x1	Start address x4	End address x4	LTDC common	LTDC_L1L2	LTDC_L3
LTDC	Common CFG0	0x0	0x0	0x30	Y	-	-
	Interrupt-A	0xD	0x34	0x3C	-	Y	-
	Common CFG1	0x10	0x40	0x60	Y	-	-
	Interrupt-B	0x19	0x64	0x70	-	-	Y
	(empty)	0x1D	0x74	0x74	Y	-	-
	CRC	0x1E	0x78	0x7C	-	-	Y
	(empty)	0x20	0x80	0x88	Y	-	-
	Common CFG2	0x23	0x8C	0xFC	Y	-	-
	L1-CFG0 (typ: video)	0x40	0x100	0x100	-	Y	-
	L1-CTRL	0x41	0x104	0x104	-	Y	-
	L1-CFG1	0x42	0x108	0x1FC	-	Y	-
	L2-CFG0 (typ: UI)	0x80	0x200	0x200	-	Y	-
	L2-CTRL	0x81	0x204	0x204	-	Y	-
	L2-CFG1	0x82	0x208	0x2FC	-	Y	-
	L3-CFG0 (typ: secu)	0xC0	0x300	0x300	-	-	Y
	L3-CTRL	0xC1	0x304	0x304	-	-	Y
	L3-CFG1	0xC2	0x308	0x3FC	-	-	Y

45.5 Display subsystem programmable parameters

Steps to configure the various main use cases supported by the display subsystem are detailed below. They provide the global picture and settings of the display subsystem (not repeating information available in the local sections).

45.5.1 Parallel-interface display

This is the default use case to drive a display panel or bridge through the parallel interface. It allows panels to be driven up to 1080p60 (150 Mpixel/s), with either RGB888 or YUV422.

The RCC provides a pixel clock that drives both the LTDC and its synchronous parallel interface. Aside, the LTDC needs an AXI-Master clock to fetch pixels, and an APB-Slave clock to get configured.

The steps to follow are detailed below:

1. Setup security settings, with or without a secure layer depending on application needs, as described in *Figure 387*.
2. Clock and reset:
   - ck_ker_ltdc clock, provided by the RCC, as output pixel clock.

     One PLL of the RCC needs to be allocated to the display subsystem to provide the clock that is exactly at the wanted pixel rate for the output display. For instance, a standard 1080p60 requires a 148,50 MHz clock.
   - DISPLAYCLKCR(1:0) setting = 0x2, to select the above ck_ker_ltdc as input for the pixel_clk of the LTDC.
   - o_clk_display_parallel, is driven automatically by the LTDC (after some gating) from its above provided pixel_clock (from the RCC ck_ker_ltdc).
   - ck_icn_p_ltdc is provided by the RCC to allow the LTDC configuration (default 150 MHz).
   - ck_icn_m_ltdc is provided by the RCC to allow the LTDC AXI to work (default 400 MHz).

   See *Section 20: Reset and clock control (RCC)* for more details.
3. LTDC configuration:
   - Setup the LTDC (such as input layers, display information, parallel interface, AXI settings) as described in the LTDC section.
4. Signal polarity:

   The polarities of the signals are impacted by the fact that the LTDC parallel interface signals are reclocked in the IO-ReTime logic, downstream of the LTDC. It results the following when configuring the polarities to match the panel on the parallel interface:
   - The clock polarity (rising-vs-falling edge) must be configured in the IO-ReTime only (see *Section 22: General-purpose I/Os (GPIO)*), and not in the LTDC output logic. The clock polarity logic in the LTDC must remain in its default post-reset state.
   - The polarity of the Vs, Hs, and De synchronization signals, unlike for the clock polarity, can be configured directly in the LDTC output stage (see *Section 46: LCD-TFT display controller (LTDC)*).
5. I/O configuration:

   Allocate an alternate GPIO bank to the display subsystem, with usually 28 pins needed: 24 pins for RGB888 and 4 others for the ancillary: HSync, VSync, DataEnable, and clock.

45.6 Display subsystem interrupts

The interrupts are described in sections of peripherals in this reference manual.

45.7 Display subsystem registers

All registers are described in *Section 46: LCD-TFT display controller (LTDC)*.

46 LCD-TFT display controller (LTDC)

46.1 LTDC introduction

The LTDC (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (Red, Green, Blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD-TFT panels.

46.2 LTDC main features

- 3 input layers blended together to compose the display
- Cropping of layers from any input size and location
- Multiple input pixel formats:
  - Predefined ARGB formats: ARGB8888, ABGR8888, RGBA8888, BGRA8888, RGB565, BGR565, and RGB888packed
  - Flexible ARGB, allowing any width and location for A,R,G, and B components
  - Predefined YUV formats: YUV422-1L (FourCC: YUYV, Interleaved), YUV420-2L (FourCC: NV12, semi planar), YUV420-3L (FourCC: Yxx I420, full planar) with some flexibility on the component sequence
- Color look-up table (CLUT) up to 256 colors (256 x 24 bits) per layer
- Color transparency keying
- Upscaler (horizontal and vertical) with bilinear filtering, up to 8 x 8 with decimal ratios
- Composition with flexible window position and size versus output display
- Blending with flexible layer order and alpha value (per pixel or constant)
- Background underlying color
- Gamma with non-linear configurable table
- Dithering for output with less bits per component (pseudo-random on 2 bits)
- Polarity inversion for HSync, VSync, DataEnable outputs, and clock via the I/O
- Output as RGB888 24 bpp or YUV422 16 bpp (interleaved)
- Secure layer (using layer3) capability, with grouped regs and additional interrupt set
- Interrupts based on 7 different events, controlled by two independent register sets
- AXI master interface with long efficient bursts (64 or 128 bytes)

Note: The above mentioned FourCC pixel formats are described at www.fourcc.org.

46.3 LTDC implementation

The LTDC has been implemented with the following settings:

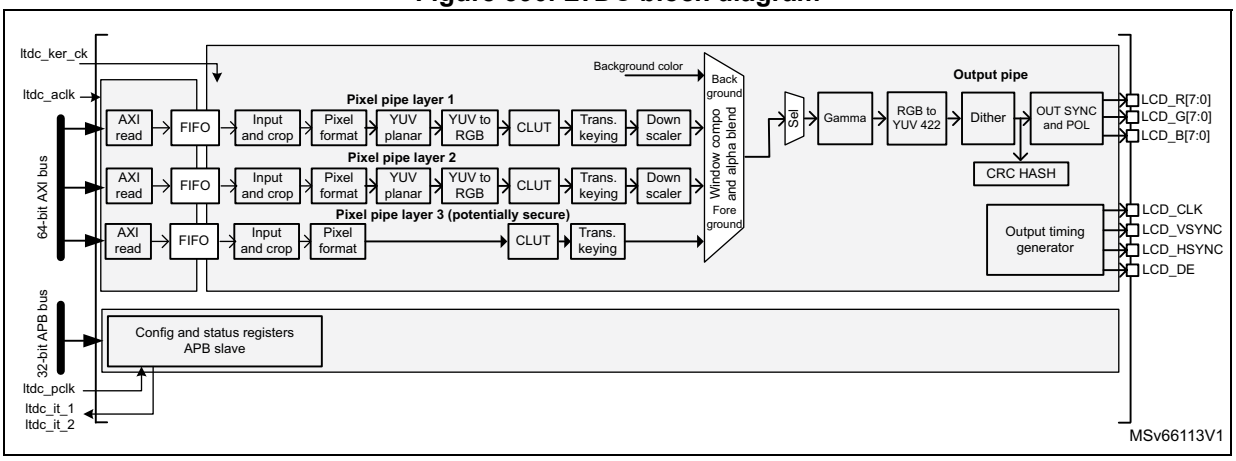
- Up to 24-bits 150 MHz pixel clock (up to 1080p60)
- AXI bandwidth: an average of up to 6 bytes per AXI cycle (2.4 Gbyte/s if 400 MHz)
- Maximum memory latency: 5 µs resilient (for 1080p60, thus 150 Mpixel/s)

These settings depend on the configured frequencies of the AXI and pixel clocks.

46.4 LTDC functional description

46.4.1 LTDC block diagram

Figure 390. LTDC block diagram



Blocks shown in the above figure are the following:

- AXI read/write: AXI DMA engines, exposing a final single read/write AXI interface
- FIFO: per-layer FIFO, sized to support large memory latencies
- Input and crop: crops any input window size from the input buffer
- Pixel format: pixel format converter, that extract 32-bit words into ARGB pixels
- YUV-to-RGB: YUV-Planar handling and YCbCr-to-RGB color conversion
- CLUT: color look-up table (256 entries) to allow indexed colors
- Transp. keying: extracts one color to define it as transparent
- Upscaler: upscaler with bilinear filtering, with decimal ratios
- Alpha blender: blending with configurable layer level and a background color
- Gamma: gamma correction, with 8-segment interpolations
- Dithering: reduces component dynamic, with pseudo-random smoothing
- RGB-to-YUV: output conversion to YCbCr, to support low-bitrate YUV422 links
- LCD-OUT: output pixel interface, with Clk, VSync, HSync, DE, RGB888
- CRC hash: CRC computation, per frame, to sign each frame and help debug
- Timing generator: handles the HSync/VSync timings and blanking
- APB registers (control, status, interrupts): accessed by APB

These operators and their configuration are described from *Section 46.5.1* to *Section 46.5.21*.

46.4.2 LTDC pins and internal signals

Table 426. LTDC pins

Pin name	Pin type	Description
LCD_CLK	Output	Clock output
LCD_HSYNC		Horizontal synchronization
LCD_VSYNC		Vertical synchronization
LCD_DE		Not data enable
LCD_R[7:0]		8-bit Red data (or U/V component, if output is YUV422)
LCD_G[7:0]		8-bit Green data (or Y component if output is YUV422)
LCD_B[7:0]		8-bit Blue data

The LTDC pins must be configured by the user application. The unused pins can be used for other purposes.

Using less pins than the default 24 bpp

The LTDC outputs the pixels on 24 pins by default (RGB888). The user can however connect the display panel using less wires, for instance 16 wires (if RGB565) or 18 wires (RGB666). If so, R (resp. G, B) wires must be connected to the MSB of the LTDC R (resp. G, B) output pins (see *Section 46.4.2*).

Example: LTDC interfacing with a RGB565 16-bit display panel. Display panel R[4:0], G[5:0] and B[4:0] wires and input pins must be connected to LTDC_R[7:3], LCD_G[7:2], and LCD_B[7:3] output pins.

46.4.3 LTDC reset and clocks

The LTDC uses the following clock domains:

- **AXI clock domain** (see ck_icn_m_ltdc in *Section 20: Reset and clock control (RCC)*)
  This domain contains the LTDC AXI master interface for data transfer from the memories to the layer FIFO and the frame buffer configuration register.
- **APB clock domain** (see ck_icn_p_ltdc in *Section 20: Reset and clock control (RCC)*)
  This domain contains the global configuration registers and the interrupt register.
- **Pixel clock domain** (see *Section 45: Display subsystem*)
  This domain contains the pixel data generation, the layer configuration register as well as the LTDC interface signal generator. The LCD_CLK output must be configured following the panel, or bridge (HDMI) requirements.

Note: For normal operation, all clocks must be provided to allow clock domain crossing for read and write operations.

The following table lists the clock domains used to store the register values.

Table 427. Clock domain for each register

LTDC register	Clock domain
LTDC_IDR	APB clock
LTDC_LCR	
LTDC_SSCR	Pixel clock
LTDC_BPCR	AXI + pixel clock
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_GC1R	APB clock
LTDC_GC2R	
LTDC_SRCR	
LTDC_GCCR	Pixel clock
LTDC_BCCR	
LTDC_IER	APB clock
LTDC_ISR	
LTDC_ICR	
LTDC_LIPCR	Pixel clock
LTDC_CPSR	
LTDC_CDSR	
LTDC_EDCR	
LTDC_IER2	APB clock
LTDC_ISR2	
LTDC_ICR2	
LTDC_LIPCR2	Pixel clock
LTDC_ECRCR	
LTDC_CCRCR	
LTDC_RB0AR	AXI clock
LTDC_RB1AR	
LTDC_RBPR	
LTDC_RIFCR	
LTDC_FUTR	Pixel clock
LTDC_LxC0R	APB clock
LTDC_LxC1R	
LTDC_LxRCR	

Table 427. Clock domain for each register (continued)

LTDC register	Clock domain
LTDC_LxCR	AXI + pixel clock
LTDC_LxWHPCR	Pixel clock
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxBLCR	AXI clock
LTDC_LxPCR	AXI + pixel clock
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_LxAFBA0R	
LTDC_LxAFBA1R	
LTDC_LxAFBLR	
LTDC_LxAFBLNR	
LTDC_LxCLUTWR	Pixel clock
LTDC_LxSISR	
LTDC_LxSOSR	
LTDC_LxSVSFR	
LTDC_LxSVSPR	
LTDC_LxSHSFR	
LTDC_LxSHSPR	
LTDC_LxCYR0R	
LTDC_LxCYR1R	
LTDC_LxFPF0R	
LTDC_LxFPR1R	

While accessing LTDC registers, the APB bus is stalled during the access for a given time period detailed in the following table.

Table 428. LTDC register access and updated durations

Access type	AXI register clock domain	APB register clock domain	Pixel register clock domain
Read	7 x ltdc_pclk + 5 x ltdc_aclk	7 x ltdc_pclk	7 x ltdc_pclk + 5 x ltdc_ker_clk
Write	6 x ltdc_pclk + 5 x ltdc_aclk	6 x ltdc_pclk	6 x ltdc_pclk + 5 x ltdc_ker_clk

The LTDC can be reset by setting the corresponding bit in the RCC, that resets the three clock domains.

46.5 LTDC configuration parameters

The LTDC configuration parameters are sequenced following the pixel flow as shown in *Figure 390*, starting at the input (AXI master that reads pixels) and ending at the pixel output.

Common controls (such as interrupts) are listed at the end.

46.5.1 AXI master

The AXI master busses perform the accesses to the memory and are driven by the internal logic. The LTDC internal per-layer FIFOs are fairly large to cope with reasonably long external memory latencies.

The LDTC generates long burst (128-byte) for higher external memory efficiency.
For more granularity, it is possible to decrease the max burst size to 64 bytes, but it is not recommended.

Parameter in LTDC_LxBLCR

BL: burst length, in 64-bit words, with a default 0 for max size

Default configuration

The default AXI burst length is set at its maximum value after reset, with BL = 0.

46.5.2 Input layer definition and cropping

The LTDC supports three layers. Each layer is defined by:

- Its base-address in the memory

  If the buffer is YUV semi-planar or full-planar, there are respectively two and three base-addresses. The first buffer is called main and stores the luminance data, while the other buffers are called auxiliary and store the chrominance data.
- A dimension in the memory
  - Width defined as its pitch (or address jump or stride): amount of bytes between the start of one line and the start of the next line
  - Height that does not need to be defined
- A dimension of the pixel rectangle that is cropped and read from the memory

  - Top-left start position: set by the base-address at exactly the byte where the top-left pixel is located
  - Width, number of bytes per line to read
    - If too few bytes versus required, a FIFO underrun interrupt is generated(FUWIF).
    - If too many bytes versus required, the useless data are discarded.
  - Height, number of lines per frame to read

Note: The pitch and number of bytes per line can be different if a sub-part of a frame is cropped to be displayed. If the full frame is displayed, the pitch and number of lines are the same.

Parameters

- LEN in LTDC_LxCR: activates this layer.
- CFBADD in LTDC_LxCFBAR: main base address, for ARGB or Y buffer
- CFBLL in LTDC_LxCFBLR: main number of bytes per line
- CFBP in LTDC_LxCFBLR: pitch between two consecutive lines of the main buffer
- CFBLNBR in LTDC_LxCFBLNR: main number of lines per frame

Specifically for multi-planar-capable layers, the following additional registers have to be configured (only required for YUV420 semi-planar and full-planar pixel formats):

- AFBADD0 in LTDC_LxAFBA0R: auxiliary address 0, for UV if semi-planar or for V if full-planar
- AFBADD1 in LTDC_LxAFBA1R: auxiliary address 1, for U if full-planar
- AFBLL in LTDC_LxAFBLR: auxiliary 0 and 1, number of bytes per line
- AFBP in LTDC_LxAFBLR: pitch for the auxiliary 0 and 1 buffers
- AFBLNBR in LTDC_LxAFBLNR: auxiliary 0 and 1 number of lines per frame

Configuration examples

- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.
- A frame buffer having the format CbCr (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format Cb (1 byte per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 1 = 320), where pitch = line length requires a value of 0x01400147 to be written into this register.

46.5.3 Input pixel format

The LTDC reads the pixel data from memory buffers and maps it onto the internal LTDC format that is based on the definition of the input pixel format, defined per layer.

The pixel formats are grouped as follows:

- **Predefined ARGB formats:**

  ARGB8888, ABGR8888, RGBA8888, BGRA8888, RGB565, BGR565, and RGB888pack
- **Flexible ARGB format:**

  Allows a configurable width and location of each A,R,G, and B component and is used to extend the above list to any multi-component pixel format except to coplanar ARGB.
- **YUV formats:**

  Used to read YUV with a 422 coplanar, 420 semi-planar, or 420 full-planar format, and to convert the YUV color into RGB.

  The YUV option is not available for the layer 3.

Note: The LTDC hardware pipeline uses 8 bits per component: when reading and mapping a pixel format, with less than 8 bits per component. That component is extended to 8 bits by stuffing the LSB, that are recopied from its MSB. This applies mostly on RGB565.

For instance, a component on 5 bits (named: 43210) results in the following 8 bits mapped into the pipe: 43210432. A 3-bit input (named 210) results in 21021021.

Parameter when using the predefined ARGB formats

- PF in LTDC_LxPFCR: 0 to 6 to select one of the seven predefined pixel formats

Parameters when using the flexible ARGB formats

- PR in LTDC_LxPFCR: seven to activate the flexible pixel format type
- PSIZE in LTDC_LxFPF1R: defines the amount of bytes per pixel (typically 1,2,3, or 4)
- APOS/RPOS in LTDC_LxFPF0R: defines the Alpha/Red location inside a 32-bit word
- BPOS/GPOS in LTDC_LxFPF1R: defines the Blue/Green location inside a 32-bit word
- ALEN/RLEN in LTDC_LxFPF0R: defines the amount of bits for Alpha/Red components
- BLEN/GLEN in LTDC_LxFPF1R: defines the amount of bits for Blue/Green components

  Examples:

  An RGB565 pixel format is defined as follows:
  - Alpha: APOS = 0, ALEN = 0
  - Red: RPOS= 11, RLEN = 5
  - Green: GPOS = 5, GLEN = 6
  - Blue: BPOS = 0, BLEN = 5

  The BGRA8888 32 bpp pixel format is set as follows (default reset values of LTDC_LxFPF0R and LTDC_LxFPF1R):
  - Alpha: APOS = 0, ALEN = 8
  - Red: RPOS= 8, RLEN = 8
  - Green: GPOS = 16, GLEN = 8
  - Blue: BPOS = 24, BLEN = 8

Parameters when using the YUV formats

The YUV pixel format is defined and detailed in *Section 46.5.4: YUV planar* and supersedes any setting here.

The following table describes the memory mapping of the predefined ARGB pixel format.

Table 429. Pixel data mapping versus color format

ARGB8888 (PF = 000)			
@+3 A_x[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 A_{x+1}[7:0]	@+6 R_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 B_{x+1}[7:0]
ABGR8888 (PF = 001)			
@+3 A_x[7:0]	@+2 B_x[7:0]	@+1 G_x[7:0]	@ R_x[7:0]
@+7 A_{x+1}[7:0]	@+6 B_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 R_{x+1}[7:0]
RGBA8888 (PF = 010)			
@+3 R_x[7:0]	@+2 G_x[7:0]	@+1 B_x[7:0]	@ A_x[7:0]
@+7 R_{x+1}[7:0]	@+6 G_{x+1}[7:0]	@+5 B_{x+1}[7:0]	@+4 A_{x+1}[7:0]
BGRA8888 (PF = 011)			
@+3 B_x[7:0]	@+2 G_x[7:0]	@+1 R_x[7:0]	@ A_x[7:0]
@+7 B_{x+1}[7:0]	@+6 G_{x+1}[7:0]	@+5 R_{x+1}[7:0]	@+4 A_{x+1}[7:0]
RGB565 (PF = 100)			
@+3 R_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] B_{x+1}[4:0]	@+1 R_x[4:0] G_x[5:3]	@ G_x[2:0] B_x[4:0]
@+7 R_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] B_{x+3}[4:0]	@+5 R_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] B_{x+2}[4:0]
BGR565 (PF = 101)			
@+3 B_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] R_{x+1}[4:0]	@+1 B_x[4:0] G_x[5:3]	@ G_x[2:0] R_x[4:0]
@+7 B_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] R_{x+3}[4:0]	@+5 B_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] R_{x+2}[4:0]
RGB888 (PF = 110)			
@+3 B_{x+1}[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 G_{x+2}[7:0]	@+6 B_{x+2}[7:0]	@+5 R_{x+1}[7:0]	@+4 G_{x+1}[7:0]
The flexible format (PF = 111), not represented here, allows any above mapping.			

46.5.4 YUV planar

This operator handles the YUV planar formats, reading data from potentially several distinct buffers (Y versus Cb versus Cr) into a single pixel information. The feature is available for layer 1 and layer 2. It is not available for layer 3.

The LTDC supports the following planar formats:

- YUV422 coplanar (one buffer)
- YUV420 semi-planar (two buffers)
- YUV420 full-planar (three buffers)

Following its configuration, the LTDC reads the relevant luminance and chrominance buffers, upscales the chrominance components required to reach YUV444 pixels, and converts the color from YUV to RGB as described in *Section 46.5.5*.

The LTDC allows swapping the memory locations of the Y and Cb/Cr data and of the Cb and Cr data. It also allows inverting the endianness of the Y data. This extends the amount of supported pixel format variants.

Parameters

Pixel format parameters are stored in LTDC_LxPCR:

- YCEN: activates the YUV mode (both the planar extraction and color conversion).
- YCM: selects the input scheme (422 for coplanar, 420 for semi-planar, 420 for full-planar).
- YF: defines if either Y or Cb/Cr is provided first (from LSB).
- CBF: defines if either Cb or Cr is provided first (from LSB).
- OF: defines if either odd or even Y is provided first (from LSB).

When using a semi-planar or full-planar buffer, some additional base-address must be configured for auxiliary (chrominance) buffers, as well as the pitch of these buffers (see *Section 46.5.2*).

Configuration examples

Settings of OF, CBF, YF bits for a YCbCr 422 pixel format:

- OF = 0, CBF = 0, YF = 0 define the buffer word to containWord[31:0] = Y1-Cb-Y0-Cr.
- OF = 1, CBF = 0, YF = 0 define the buffer word to contain Word[31:0] = Y0-Cb-Y1-Cr.
- OF = 0, CBF = 1, YF = 0 define the buffer word to contain Word[31:0] = Y0-Cr-Y1-Cb.
- OF = 0, CBF = 0, YF = 1 define the buffer word to contain Word[31:0] = Cb-Y1-Cr-Y0.
- OF = 1, CBF = 1, YF = 1 define the buffer word to contain Word[31:0] = Cr-Y0-Cb-Y1.

  These settings are also valid for YCbCr 420 pixel formats.

46.5.5 YUV-to-RGB color conversion

This operator handles the YUV-to-RGB color conversion of the YUV planar pixels. Like the above operator, it is only available for layer 1 and layer 2. The color conversion is activated automatically, simultaneously with the YUV planar handling.

The color conversion is based on a semi-flexible 3x3 matrix multiplication:

- It is restricted to YUV-to-RGB conversion due to some frozen coefficients (see next).

- It handles the most used YUV references (ITU-R BT.601 and ITU-R BT.709) thanks to some configurable coefficients.
- It supports the full versus reduced dynamic of the input pixel (luminance defined either in 0 to 255, or in 16 to 235), by extending a potential reduced-range input to a full range.

Parameters

- YCEN in LTDC_LxPCR: activates the color conversion, same bit as for planar.
- YREN in LTDC_LxPCR: if active, extends a reduced range to a full range.
- CR2R in LTDC_LxCYR0R: matrix coefficient from Cr to Red
- CB2B in LTDC_LxCYR0R: matrix coefficient from Cb to Blue
- CR2G in LTDC_LxCYR0R: matrix coefficient from Cr to Green
- CB2G in LTDC_LxCYR0R: matrix coefficient from Cb to Green

The full matrix computation, to convert from YCbCr to RGB, is defined as the following:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} CR2R & 256 & 0 \\ -CR2G & 256 & -CB2G \\ 0 & 256 & CB2B \end{pmatrix} \times \begin{pmatrix} Cr - 128 \\ Y \\ Cb - 128 \end{pmatrix}$$

The CR2R, CB2B, CR2G, and CB2G coefficients are defined on 10 bits unsigned: as the operator works with 8-bit components, the coefficients are thus equivalent, in fixed-point, to 2-bit integer and 8-bit decimals.

The CR2G and CB2G coefficients are configured as unsigned positive, but are set negative in the hardware computation, as shown in the above matrix where a “-” is mentioned.

When the luminance scaling is active (YREN = 1, thus where the YCbCr input is assumed in the range of 16 to 235/240, and RGB in 0 to 255), then the Y input value is first scaled in hardware to Y’ = 255/219 * (Y - 16). That first result is then inserted as input to the above matrix multiplication.

Configuration examples

To read an ITU-R BT.601 layer and to convert its YCbCr (assuming a default reduced dynamic in 16 to 235/240) to RGB (with full dynamic in 0 to 255), the recommended coefficients are:

- YCEN = 1: to activate the conversion
- YREN = 1: to select the reduced dynamic range
- CR2R = 409: specific coefficient set for ITU-R BT.601
- CR2G = 208
- CB2G = 100
- CB2B = 516

To read an ITU-R BT.709 layer and to convert its YCbCr (assuming a default reduced dynamic in 16 to 235/240) to RGB (with full dynamic in 0 to 255), the recommended values to configure coefficients are:

- YCEN = 1: to activate the conversion
- YREN = 1: to select the reduced dynamic range
- CR2R = 459: specific following coefficient set is for ITU-R BT.709
- CR2G = 136
- CB2G = 55
- CB2B = 541

46.5.6 Horizontal or vertical mirroring

The LTDC can mirror horizontally or vertically its input layers, thus to read them from right to left, and/or from bottom to top.

It applies on both coplanar (ARGB, YUV422) and multi-planar (YUV420) pixel formats.

Parameters for horizontal mirroring

To mirror a layer horizontally, the LTDC must be configured to extract the pixels from right to left (backward), and to set the layer base-address at the right-end of the line.

The configuration bitfields are the following:

- HMEN in LTDC_LxCR: activates the horizontal mirroring.
- CFBADD in LTDC_LxCFBAR = base + (width - 1) (increase for horizontal mirror)
- AFBADD0LTDC_LxAFBA0R = base + (width - 1)
- AFBADD1 in LTDC_LxAFBA1R = base + (width - 1)

Parameters for vertical mirroring

To mirror a layer vertically, the LTDC must be configured to start reading this layer from the bottom, and line per line, to come upward. The layer base-address must be set to the start of the bottom line, and a negative line-per-line pitch must be provided.

The configuration bitfields for the base-address are (all to be set at bottom line start):

- CFBADD in LTDC_LxCFBAR: main base-address
- AFBADD0 in LTDC_LxAFBA0R: auxiliary address 0
- AFBADD1 in LTDC_LxAFBA1R: auxiliary address 1
- CFBADD in LTDC_LxCFBAR = base + (lines - 1) * pitch (increase for vertical mirror)
- AFBADD0 in LTDC_LxAFBA0R = base + (lines - 1) * pitch
- AFBADD1 in LTDC_LxAFBA1R. = base + (lines - 1) * pitch

The configuration bitfields for the pitch are (all to be set with a negative pitch):

- CFBP in LTDC_LxCFBLR: -pitch (negative line pitch for vertical mirror)
- AFBP in LTDC_LxAFBLR: - pitch

46.5.7 Default layer color

Every layer has a default color in the format ARGB8888, that is used outside the defined layer window or when a layer is disabled. This default color can be configured as active or absent.

Parameters

The default layer color parameters are:

- DCBEN in LTDC_LxCR: default color activation
- DCALPHA in LTDC_LxDCCR: default Alpha transparency setting
- DCRED in LTDC_LxDCCR: default Red component
- DCGREEN in LTDC_LxDCCR: default Green component
- DCBLUE in LTDC_LxDCCR: default Blue component

Default configuration

By default, and after reset, the default layer color is inactive (DCBEN = 0 and DCALPHA = 0.

Note: The blending of a layer is always performed even when a layer is disabled. To avoid displaying the default color when a layer is disabled, the default Alpha must be set to null (DCALPHA = 0), or the whole default color must be disabled (with DCBEN).

46.5.8 Color look-up table (CLUT)

The LTDC supports frame buffers with each pixel defined as an index to a color table, allowing a display with a set of 256 colors, but stored in memory with only 8 bpp.

The CLUT defines 256 colors and must be indexed with a maximum 8-bit pixel format (typically L8 or AL44). Such format can be extracted using the flexible RGB approach described in *Section 46.5.3*.

The CLUT table is differentiated for each layer. It must be initialized with one RGB888 value per color index, with the color index corresponding to the CLUT address:

- For L8, the CLUT must be loaded with 256 colors.
- For AL44, the CLUT must be only loaded with 16 colors. Each color address is defined by replicating the 4-bit L channel to 8 bits as follows:
  - Color L4 = 0: looks out its RGB888 from index = 0x00.
  - Color L4 = 1: looks out its RGB888 from index = 0x11.
  - Color L4 = 2: looks out its RGB888 from index = 0x22.
  - ...
  - Color L4 = 15: looks out its RGB888 from index = 0xFF.
- For ARGB, the Blue component is used as the CLUT index.

Parameters

The RGB888 values and color indices are configured through the following bitfields in LTDC_LxCLUTWR:

- CLUTADD defines the color index to update.
- RED defines the 8-bit Red component of that color index.
- GREEN defines the 8-bit Green component of that color index.

- BLUE defines the 8-bit Blue component of that color index.

46.5.9 Transparency color keying

Pixels can be defined as transparent using a color key. If the pixel RGB components equals the defined RGB color key, that pixel is considered transparent and its Alpha channel is forced null.

A pixel forced to transparent is transparent when proceeding to the later blending phase, and shows the pixel of the underlying layer.

The color key is defined as an RGB888 value, that is compared with the flowing pixel after some processing in the input format operator (where LSBs are stuffed by replication) and CLUT processing. The constant color keying parameters must take these previous processing into account.

Parameters

The color keying parameters are the following:

- CKEN in LTDC_LxCR enables the color keying.
- CKRED in LTDC_LxCKCR defines the 8-bit Red color key.
- CKGREEN in LTDC_LxCKCR defines the 8-bit Green color key.
- CKBLUE in LTDC_LxCKCR defines the 8-bit Blue color key.

Configuration example

A mid-yellow color (50 % Red + 50 % Green) is used as the transparent color key. The color keying parameters take it into account as follows:

- ARGB8888 input: The mid-yellow input is 0x808000. Set LTDC_LxCKCR = 0x808000.
- RGB565 input: The mid-yellow input is 0x8400. Set LTDC_LxCKCR = 0x848200 (due to the replication of input MSB onto the missing LSB).
- CLUT input for L8, AL88 or AL44: Set one of the palette entry to the mid-yellow color 0x808000 and set LTDC_LxCKCR = 0x808000.

46.5.10 Upscaler

Layer 1 and layer 2 can be upscaled using a bilinear filter with decimal ratios. Vertical and horizontal ratios can be differentiated.

The filter is based on a bilinear filter, that provides very good results for upscale ratios from 1x to 2x, but may provide aliasing artifacts for higher ratios.

Parameters

The upscaler parameters are the following:

- SCEN in LTDC_LxCR enables the upscale.
- SIH (SIV) in LTDC_LxSISR: input width (height)
- SOH (SOV) in LTDC_LxSOSR: output width (height)
- LTDC_LxSHSFR and LTDC_LxSHSPR: horizontal ratio and phase
- LTDC_LxSVSFR and LTDC_LxSVSPR: vertical ratio and phase

The ratio is given as the input size/output size with a 16-bit fixed-point decimal. The phase is given as the start proportion of the first pixel, with a 12-bit fixed-point decimal.

Example:

To upscale from a 720p frame (1280x720 pixels) to a 1080p frame (1920x1080 pixels), the following configuration is needed:

- SCEN in LTDC_LxCR = 1 enables the upscale.
- SIH in LTDC_LxSISR = 1280: input width
- SIV in LTDC_LxSISR = 720: input height
- SOH in LTDC_LxSOSR = 1920: output width
- SOV in LTDC_LxSOSR = 1080: output height
- SHF in LTDC_LxSHSFR = 65536 * 1280 / 1920 = 43691: horizontal scaling factor
- SVF in LTDC_LxSVSFR = 65536 * 720 / 1080 = 43691: vertical scaling factor
- SHP in LTDC_LxSHSPR: horizontal phase
- SVP in LTDC_LxSVSPR: vertical phase

Phase can be kept by default at 0, to start the upscaled layer with the left-most top-most as a native pixel (recopy from the input)

46.5.11 Display composition - windowing

When performing the display composition, every layer (after a potential upscale) can be cropped and positioned into the active display area. The window position and size are configured by defining its top-left and bottom-right corners.

Note: The locations are defined versus the internal timing generator that must take into account the potential synchronization header, the back porch size and the chosen active pixel area (as shown in Figure 391).

The mapping between the input layer and the visible window is performed as follows:

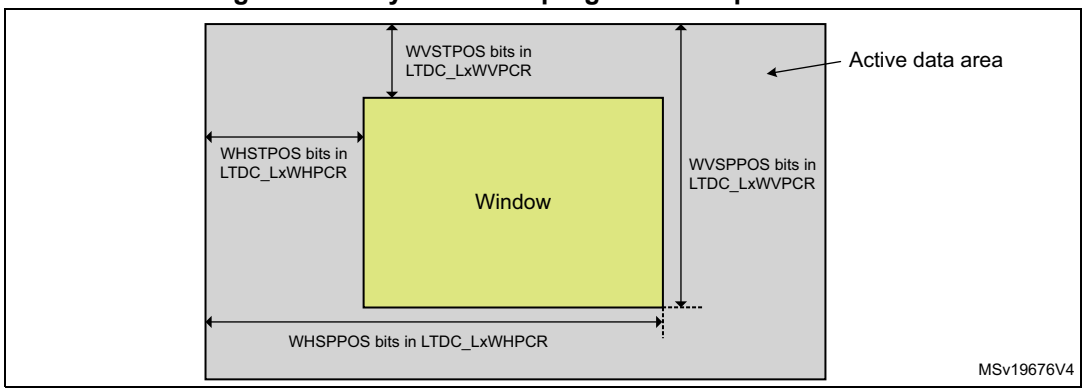
- The most top-left pixels of the input layer are mapped to the top-left of the visible window of this layer.
- If the visible window is smaller (less wide, less high) than the input layer, then the exceeding input pixels are automatically discarded from the pipe.

Parameters:

The following parameters are defined versus the top-left of the synchronization:

- WHSTPOS in LTDC_LxWHPCR: first (left) visible pixel of the layer
- WHSPPOS in LTDC_LxWHPCR: last (right) visible pixel of the layer
- WVSTPOS in LTDC_LxWVPCR: first (top) visible line of the layer
- WVSPPOS in LTDC_LxWVPCR: last (bottom) visible line of the layer

Figure 391. Layer window programmable parameters



Configuration examples

- LTDC_BPCR is configured to 0x000E0005 (AHBP[11:0] is 0xE), and LTDC_AWCR is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:
  - Layer window first pixel: WHSTPOS[11:0] must be programmed to 0x14 (0xE + 1 + 0x5).
  - Layer window last pixel: WHSPPOS[11:0] must be programmed to 0x28A.
- LTDC_BPCR is configured to 0x000E0005 (AVBP[11:0] is 0x5) and LTDC_AWCR is configured to 0x028E01E5 (AAH[11:0] is 0x1E5). To configure the vertical position of a window size of 630x460, with vertical start offset of eight lines in the active data area:
  - Layer window, first line: WVSTPOS[11:0] must be programmed to 0xE (0x5 + 1 + 0x8).
  - Layer window last line: WVSTPOS[11:0] must be programmed to 0x1DA.

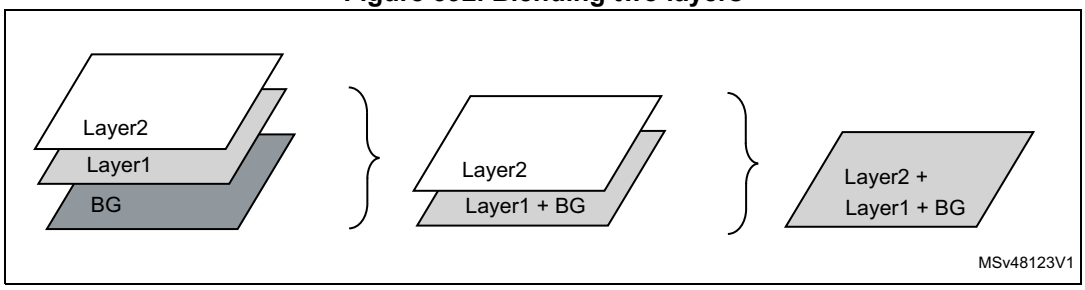
46.5.12 Display composition - blending

The Alpha blending stage is used to blend together the 3 input layers, on top of a background color.

It is possible to select the sequence of the layers (foreground, middle, and background), to apply a global multiplicative Alpha per-layer, and to define an underlying background color.

The blending computation is performed from back to front, sequentially, as shown the following figure, with only two layers for the example instead of three.

Figure 392. Blending two layers



The general formula for a 2-to-1 blending is CR = (BF1 x C1) + (BF2 x C2) where:

- CR: resulting blended color
- BF1: blending-factor of current layer
- C1: layer-color of the current layer
- BF2: blend-factor of underlying pixels
- C2: layer-color of the underlying pixels

Both blending factor, BF1 and BF2, can be defined as one of the following:

- Pure per-pixel Alpha value
- Mix of per-pixel Alpha and a global per-layer Alpha (the mix is multiplicative)

Parameters

- CONSTA in LTDC_LxCACR: global multiplicative Alpha value of the current layer
- BOR in LTDC_LxBFCR: blending order of the current layer
- BF1 in LTDC_LxBFCR: blending factor for the current layer
- BF2 in LTDC_LxBFCR: blending factor for the underlying layer

The background color is defined by the following bitfields in LTDC_BCCR:

- BCRED: background Red component
- BCGREEN: background Green component
- BCBLUE: background Blue component

Note: For secure purposes, in case of an inconsistent configuration, the blending order falls back to a fixed order, with the layer 3 on the foreground, the layer 2 in middle, and layer 1 in background, followed by the underlying background color.

The inconsistent fallback is setup to protect a potential secure layer: If a secure software configures the secure layer3 to be on the foreground, and a malware tries to place that another layer on the foreground, the inconsistency is detected, and the secure layer is automatically forced on the foreground, guaranteeing that it is visible.

Configuration example

Only layer 1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in LTDC_LxCACR is 240 (0xF0). Thus, the constant alpha value is 240 / 255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer 1 is blended with the background color.
BC = constant alpha x C + (1 - constant alpha) x Cs = 0.94 x 128 + (1 - 0.94) x 48 = 123.

Default configuration

All active layers can be made visible and blended by keeping the following configuration values after reset:

- CONSTA: keep as reset CONSTA = 255.
- BOR: keep as reset BOR (layer) = LayerID-1.
- BF1: keep as reset BF1 = 6 for default mixed Alpha.
- BF12: keep as reset BF2 = 7 for default mixed 1-Alpha.
- Background color: keep as reset (black), although any color is ok.

46.5.13 Gamma correction

The gamma correction is a non-linear operator that adapts the pixel to the non-linear display, using a compensating pre-distortion. The R, G, B components can be independently adapted, using a configurable non-linear transformation.

The non-linear transformation is implemented in the LTDC as eight independent linearly-interpolated segments: the desired non-linear transformation must thus be mapped onto the eight segments, and the shape of the eight segments configured in LTDC registers.

Parameters

- GAMEN in LTDC_GCR: activation of the gamma operator
- ADDR in LTDC_GCCR: table configuration, index (among 256) to write-in
- COMP in LTDC_GCCR: table configuration, 8-bit component to write-in
- REN in LTDC_GCCR: table configuration, active to write into the Red table
- GEN in LTDC_GCCR: table configuration, active to write into the Green table
- BEN in LTDC_GCCR: table configuration, active to write into the Blue table

Note: As the gamma correction works with eight interpolated segments. The only table indices to write-in are located at index: 0, 32, 64, 96,...,224, 255. The value to provide for index 255 contains the value for the pseudo-index 256.

Example configuration

The gamma correction is recommended to be set to pre-compensate the pixels with a gamma exponent of 2.2: Red, Green, and Blue components are compensated the same way by the Gamma. The configuration of Red, Green, and Blue tables is then usually the same.

Default configuration

The gamma correction is disabled with the default configuration after reset, with GAMEN = 0.

46.5.14 YUV output conversion

The default RGB888 24 bpp output can be converted to a YUV422 16 bpp output (coplanar), in order to reduce the bitrate or the interface width, if necessary.

The operator can convert the RGB to YUV with either ITU-R BT.601 versus BT.709 reference, and filter down the chrominance, to output the pixels as a YUV422 interface.

When the conversion is enabled, the Y is output on the LCD_G (green bits) and the Cb/Cr are output on the LCD_R (red bits).

Parameters in LTDC_EDCR

- OCYEN: activates the output conversion to YUV422.
- OCSEL: selects between a color conversion to ITU-R BT.601 versus to BT.709.
- OCYCO: defines whether Cb or Cr is output first.

The configuration must follow what is expected by the external display panel.

Default configuration

The YUV output conversion is disabled with the default configuration after reset, with OCYEN = 0, allowing a default RGB888 to be output.

46.5.15 Dithering

The dithering operator allows a clean conversion of the RGB888 output into a reduced pixel format, like RGB565 or RGB666, and so avoiding the usual artifact of flat area that occurs when no dithering is available.

The dithering is based on a LFSR (polynomial generating random) and adds pseudo-random noise on the LSB of each of the displayed component (bit 2 in case of an RGB666), and makes that the average of the local pixels represents the information content of the full 8-component value.

Parameters in LTDC_GCR

- DEN: dithering enable
- DRW: bit-id of LSB of the displayed Red component (among 8 bits)
- DGW: bit-id of LSB of the displayed Green component (among 8 bits)
- DBW: bit-id of LSB of the displayed Blue component (among 8 bits)

Example

To dither from RGB888 to RGB565 (assuming a RGB565 display panel):

- The LTDC outputs the Red component on pins LCD_R[7:0].
- The display panel probes the five MSB bits LCD_R[7:3].
- The red bit to dither, using its two following LSB as decimals, is the bit 3.
- DRW must be configured to 3.
- DGW = 2 and DBW = 3.

Similarly, a dithering to RGB666 requires DRW = DGW = DBW = 2.

Default configuration

The dithering is disabled with the default configuration after reset, with DEN = 0, allowing a default RGB888 to be output.

46.5.16 CRC hashing

The CRC operator allows a 16-bit signature to be extracted for each frame, by hashing the R,G,B components of all the frame pixels.

The computed CRC can be read in LTDC_CCRCR.

An expected CRC value can be configured: if the computed CRC does not match the expected CRC, the CRC interrupt is triggered. It allows a system test, letting the LTDC run for hours, and be warned only if a failed frame occurs.

Parameters

- CRCEN in LTDC_GCR: activates the CRC computation
- ECRC in LTDC_ECRCR: expected 16-bit CRC value
- CCRC in LTDC_CCRCR: computed 16-bit CRC value

Default configuration

The CRC hashing is disabled with the default configuration after reset, with CRCEN = 0.

46.5.17 Display timing

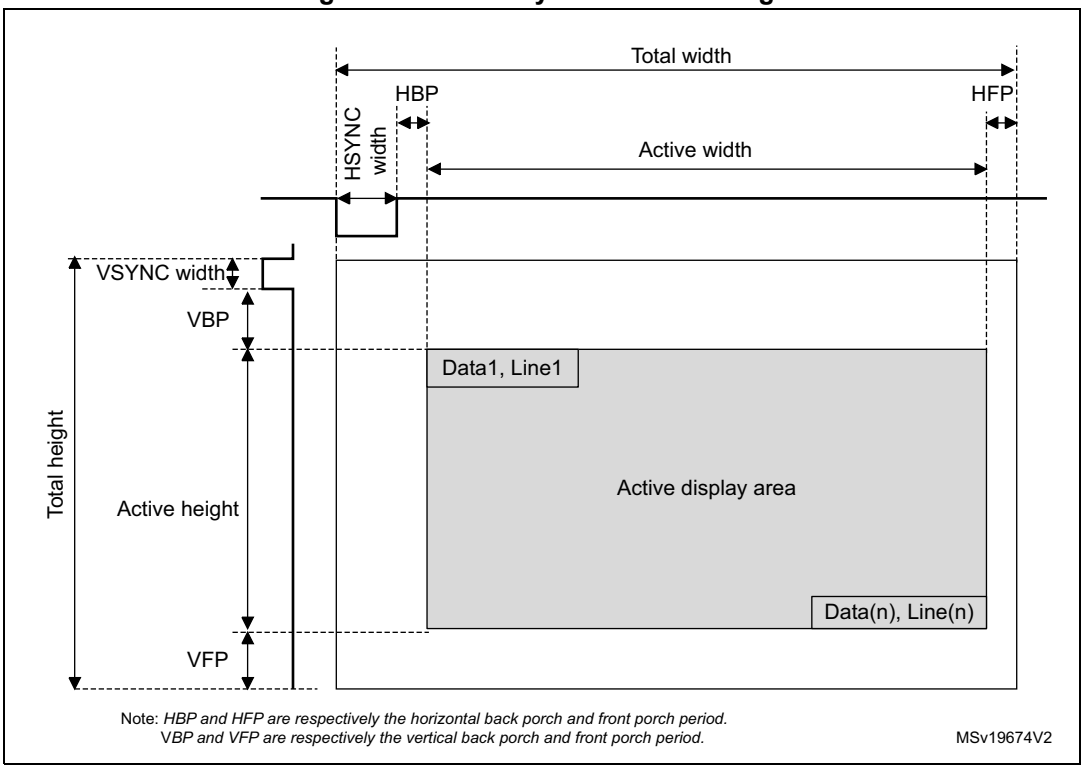
This section describes the configuration of the display timings, so that the LTDC matches what is expected by the external display panel.

The configuration defines the horizontal and vertical duration of each of the following items:

- Synchronization period (HSYNC and VSYNC)
- Back porch (HBP, VBP)
- Active display (pixels that are visible on the screen)
- Front porch (HFP, VFP)

These timings are described in the following figure.

Figure 393. LTDC synchronous timings



Note: *HBP and HFP are respectively the horizontal back porch and front porch period.*
VBP and VFP are respectively the vertical back porch and front porch period.

MSv19674V2

As a result of these timings, the LTDC samples in pixels during the active display duration, and generates the output synchronization signals (HSYNC, VSYNC), the pixel data, and the data-enable signal.

Parameters

All the timings configured next are absolute timings, computed from the display start position, and not the relative increment (or width or height) of each area.

All these bitfields indicate the last pixel of each area, and hence are subtracted 1.

- **HSYNC and VSYNC width** (horizontal and vertical synchronization width) in LTDC_SSCR:
  - HSW = HSYNC width - 1
  - VSW = VSYNC width - 1
- **HBP and VBP** (last pixel/line of the horizontal and vertical synchronization back porch) in LTDC_BPCR:
  - AHBP = HSYNC width + HBP - 1
  - AVBP = VSYNC width + VBP - 1
- **Active width and active height** (last pixel/line of the active display area) in LTDC_AWCR:
  - AAW = HSYNC width + HBP + active width - 1
  - AAW = VSYNC width + VBP + active height - 1
- **Total display width** (last pixel/line of the total display timings) in LTDC_TWCR:
  - TOTALW = HSYNC width + HBP + active width + HFP - 1
  - TOTALH = VSYNC height + VBP + active height + VFP - 1

Note: When the LTDC is enabled, timings generated start at the position (X,Y) = (0,0), and generate immediately the first horizontal synchronization pixel in the vertical synchronization, followed by the back porch, then the active display area, and finally the front porch.

When the LTDC is disabled, the timing generator block is reset to X = total width - 1, Y = total height - 1. It holds the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore, only blanking data is output continuously.

Example of synchronous timing configuration

Assume the following display panel timings (to be extracted from panel datasheet):

- Horizontal and vertical synchronization width: 0xA pixels, 0x2 lines
- Horizontal and vertical back porch: 0x14 pixels, 0x2 lines
- Active width and active height (320x240): 0x140 pixels, 0x0F0 lines
- Horizontal and vertical front porch: 0xA pixels 0x4 lines

Bitfield values to configure become:

- HSW[11:0] = 0x9 (horizontal synchro -1) in LTDC_SSCR.
- VSH[11:0] = 0x1 (vertical synchro -1) in LTDC_SSCR.
- AHBP[11:0] = 0x1D (horizontal synchro + back porch -1) in LTDC_BPCR.
- AVBP[11:0] = 0x3 (vertical synchro + back porch -1) in LTDC_BPCR.
- AAW[11:0] = 0x15D (horizontal synchro + back porch + width -1) in LTDC_AWCR.
- AAH[11:0] = 0xF3 (vertical synchro + back porch + height -1) in LTDC_AWCR.
- TOTALW[11:0] = 0x167 (horizontal synchro + back porch + width + front porch -1) in LTDC_TWCR.

- TOTALH[11:0] = 0xF7 (vertical synchro + back porch + width + front porch -1) in LTDC_TWCR.

Default configuration

There is no default configuration to propose. As display timings depend on the specific display panel and its own required timings, all these parameters must be extracted from the display panel specification.

Note: The configuration of all the above is mandatory to have the LTDC driving correctly the external display panel.

46.5.18 Output interface polarity

The polarity of the control signals of the output interface includes the clock polarity, the vertical/horizontal synchronization, and the data-enable signal.

Parameters in LTDC_GCR

- PCPOL: polarity of the output clock
- VSPOL: polarity of the vertical synchronization
- HSPOL: polarity of the vertical synchronization
- DEPOL: polarity of the data enable

Default configuration

There is not default configuration, as the configuration of the polarity depends on the display panel specifications.

The following configuration can be used to have all items working with a positive polarity:

- PCPOL = 1 (rising-edge)
- VSPOL = 1 (active high)
- HSPOL = 1 (active high)
- DEPOL = 1 (active high)

46.5.19 Shadow registers

When activated, the LTDC is expected to display continuously data, frame after frame.

The user may want to modify some settings: for example, add a new layer, modify the layer window location, or update the displayed content by updating the pointer to the buffer that is displayed.

The LTDC provides shadow registers for most of its configuration registers to allow transitions to occur cleanly synchronized during the short vertical synchronization, when nothing is displayed: while the LTDC is busy working with a copy of its configuration registers to control the ongoing display (the active set), a new version of the registers can be progressively updated by software (the shadow set). When ready, and at the exact VSync event, the shadow set is copied into the active set, ready for the next frame.

Once the shadow set is fully updated, there are two ways to transfer it in the active set:

- Immediate copy: the copy occurs immediately.
- Delayed to next VSync: the copy occurs at the next VSync.

Note: In case of delayed copy, shadow registers must not be modified again before the reload has been done.

Reading from shadow registers returns the current active value. The new written value can only be read after the reload has taken place.

The impossibility to read a temporary value in a shadow register makes impossible to perform a read-modify-write to set/reset a single bit inside a shadow register.

A register reload interrupt can be generated if enabled in LTDC_IER or LTDC_IER2.

Shadowed registers

Non-layer registers

The non-layer registers (not prefixed with LTDC_Lx) are all non-shadowed. They contain static configurations dependent mostly on the attached display panel. They do not require update during an ongoing display.

Layer registers

The layer registers are prefixed with LTDC_Lx. Most are shadowed to support a clean frame-to-frame update. The following are not shadowed, either because their shadowing would be too costly or because they are not expected to require frequent or synchronized updates:

- LTDC_LxRCR reload control
- LTDC_LxCKCR color key (assumed static)
- LTDC_LxCLUTWR CLUT (indirect access, too costly to shadow)
- All upscaler LTDC_LxSIS/SOS/SVSFR/SHSFR (assumed static)
- LTDC_LxCYR0R/R1R YUV-to-RGB color conversion (assumed static)
- LTDC_LxFPF0R/LTDC_LxFPF1R flexible pixel format (assumed static)

Parameters

The centralized reload of the shadow register is controlled with bitfields in LTDC_SRCR:

- IMR: writing 1 triggers an immediate copy.
- VBR: writing 1 triggers a copy delayed at the next blanking.

A per-layer reload control is available with bitfields in LTDC_LxRCR:

- GRMSK delegates the per-layer reload to the central reload (above).
- IMR: writing 1 triggers an immediate copy for this layer.
- VBR: writing 1 triggers a copy delayed at the next blanking.

Default configuration

For the initial LTDC configuration, when the display is inactive, the IMR mode must be used to copy the initial configuration to the active set.

When the display is active, it is recommended to use VBR to update any parameters: the parameter transition occurs then from frame-to-frame, and not inside an ongoing frame. Applying an immediate change of the LTDC configuration within a frame results in an unpredictable behavior.

46.5.20 General control

The final and mandatory step, after all the above configurations, is to trigger the display. The LTDC has two working mode:

- Continuous mode: the LTDC continuously sends frames to the display.
- Single-frame mode: the LTDC sends a single frame after a software-based trigger (generated by a software access to a register)

Parameters in LTDC_GCR

- LTDCEN: general LTDC activation
- SFEN: moves from the continuous to the single-frame mode.
- SFSWTR: software trigger, when in single-frame mode

46.5.21 Provision for a secure layer

The LTDC allows a secure layer support, with the addition of a wrapper around the LTDC that secures the access to its registers (register-per-register), and secures the access of its AXI master to the layer framebuffers.

The layer 3 can be secured thanks to the measures described in the following sub-sections.

Configuration registers

The granularity between secure versus nonsecure items is at register level:

- A secure register contains only to-be-secured bitfields.
- Nonsecure bitfields are all in open-access registers.

It allows all registers to be segregated between secure ones (with an access only by secure software) and open ones (with an access by any nonsecure software).

Interruptions

The interruption logic has been duplicated, with one secure interrupt logic (register and pin) reserved for secure software, and another open interrupt logic (second register set and second output pin) for nonsecure software.

Blending

The blending order (foreground to background) is defined in per-layer register, so that the blending order of the secure layer3 can be securely defined as foreground. If some nonsecure software defines an other layer (layer 1 or layer 2) in foreground, the integrated hardware consistency check reverses the blending order to a fixed one, forcing the secure layer to be in the foreground.

CRC

The CRC that computes a signature of the displayed frame has its configuration located in specific registers that can be defined as secure, too.

46.6 LTDC interrupts

The LTDC provides various maskable interrupts, that are grouped into two global interrupt pins: one working interrupt pin, and one error interrupt pin.

These maskable interrupts and two global output pins are duplicated for the secure software usage, resulting in a duplicated set of registers (for masking) and a duplicated set of output pin. Hence a total of four interrupt output pins: global working, global error, secure global, secure global error.

The following working interrupts on which to synchronize software are ORed together as a global working interrupt:

- Line interrupt: generated when a configured line number is reached. Its position is Configured in LTDC_LIPCR.
- Register reload interrupt: generated when a shadow-register reload is performed during the vertical blanking period.

The following error interrupts warn about some internal processing errors, and are ORed together as a global error interrupt:

- FIFO underrun warning interrupt: generated when a pixel is requested from an empty layer FIFO.
- FIFO underrun error interrupt: generated when more pixels are requested from an empty layer FIFO.
- Transfer-error interrupt: generated when an AXI bus error occurs during data transfer.
- CRC-error interrupt: generated when the CRC computed and expected are different.

The LTDC also provides an observation mean, to retrieve X and Y positions of the currently displayed pixel.

Parameters

The following nonsecure registers are used to handle the interrupts:

- LTDC_LIPCR: defines the line-number on which the line-number interrupt occurs.
- LTDC_IER demasking bit: setting each bit enables the corresponding interrupt.
- LTDC_ISR: event bit where to read if an event has occurred
- LTDC_ICR: clear bit where to write to clean an interrupt.
- LTDC_CPSR: location with X and Y of the current displayed pixel
- LTDC_FUTR: trigger from which a FIFO underrun warning becomes a FIFO underrun error, and is the amount of missed words of 64 bits

The duplicated secure registers are used to handle the interrupts (with same respective content as the nonsecure ones): LTDC_LIPCR2, LTDC_IER2, LTDC_ISR2, LTDC_ICR2.

The following table lists the bitfields of the available interrupts, and shows the four pins on which these bitfields are directed:

- Nonsecure registers drive ltdc_lo_xx interrupts. Secure registers drive ltdc_up_xx interrupts.
- Interrupts triggered by an error are suffixed with a xx_err_it. Interrupts triggered during a normal LTDC handling are suffixed with an _it only.

Table 430. LTDC interrupt requests

Registers	Interrupt event	Event bit	Enable control bit	Interrupt signal
Nonsecure registers (IER, ISR, ICR)	Line number	LIF	LIE	ltdc_lo_li
	Register reload	RRIF	RRIE	ltdc_lo_li
	FIFO underrun warning	FUWIF	FUWIE	ltdc_lo_li
	FIFO underrun error	FUIF	FUIE	ltdc_lo_err_it
	Bus transfer error	TERRIF	TERRIE	ltdc_lo_err_it
	CRC error	CRCIF	CRCIE	ltdc_lo_err_it
Secure registers (IER2, ISR2, ICR2)	Line number	LIF	LIE	ltdc_up_li
	Register reload	RRIF	RRIE	ltdc_up_li
	FIFO underrun warning	FUWIF	FUWIE	ltdc_up_li
	FIFO underrun error	FUIF	FUIE	ltdc_up_err_it
	Bus transfer error	TERRIF	TERRIE	ltdc_up_err_it
	CRC error	CRCIF	CRCIE	ltdc_up_err_it

46.7 LTDC programming procedure

The LTDC programming procedure consists to activate the clocks, to follow all the operator sequence (as listed in *Section 46.5*) from the beginning (AXI master) to the end (output interface polarity), and to configure the needed operators as described.

The minimum configuration that must be performed is the following:

1. Enable the LTDC clock in the RCC register:
   - APB slave: ck_icn_p_ltdc in the RCC
   - AXI master: ck_icn_m_ltdc in the RCC
   - Pixel clock that fits display panel requirements (resolution, blanking, frame rate)
2. Define the buffer of at least one input layer (see *Section 46.5.2*, with for example, layer base-address, size, or buffer pitch).
3. Define this layer pixel format (see *Section 46.5.3*). ARGB8888 can simply be used, depending on the stored buffer.
4. Define where this layer is inserted in the final display (see *Section 46.5.11*): typically the top-left and bottom-right corners of the display.
5. Define how this layer is blended in the final display (see *Section 46.5.12*).
6. Define timings of the display panel to drive (see *Section 46.5.17*).

7. Define polarity of the output signals (see *Section 46.5.18*).
8. Copy all above settings, once, using the immediate global reload (see *Section 46.5.19*).
9. Start the display by activating the main enable (see *Section 46.5.19*).

Note: Most layer registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. A new write to the same register overrides the previous configuration if not yet reloaded. It is also not possible to read the temporary value of a shadowed register, which makes impossible to perform read-modify-write to set/reset a single bit inside a register.

46.8 LTDC registers

46.8.1 LTDC identification register (LTDC_IDR)

Address offset: 0x000

Reset value: 0x0004 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MINVER[7:0]								REV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **MAJVER[7:0]:** LTDC major version

Bits 15:8 **MINVER[7:0]:** LTDC minor version

Bits 7:0 **REV[7:0]:** Revision

46.8.2 LDTC layer count register (LTDC_LCR)

Address offset: 0x004

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNBR[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LNBR[7:0]**: Number of layers

46.8.3 LTDC synchronization size configuration register (LTDC_SSCR)

Address offset: 0x008

Reset value: 0x0000 0000

This register defines the number of horizontal synchronization pixels minus 1, and the number of vertical synchronization lines minus 1. Refer to *Figure 393* and *Section 46.5* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VSH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HSW[11:0]**: Horizontal synchronization width (in units of pixel clock period)
This bitfield defines the number of horizontal synchronization pixel minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **VSH[11:0]**: Vertical synchronization height (in units of horizontal scan line)
This bitfield defines the vertical synchronization height minus 1. It represents the number of horizontal synchronization lines.

46.8.4 LTDC back porch configuration register (LTDC_BPCR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (HSYNC width + HBP - 1), and the accumulated number of vertical synchronization and back porch lines minus 1 (VSYNC height + VBP - 1).
Refer to *Figure 393* and *Section 46.5* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AVBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AHBP[11:0]**: Accumulated horizontal back porch (in units of pixel clock period)
This bitfield defines the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.
The horizontal back porch is the period between horizontal synchronization going inactive, and the start of the active display part of the next scan line.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AVBP[11:0]:** Accumulated Vertical back porch (in units of horizontal scan line)
This bitfield defines the accumulated vertical back porch width that includes the vertical synchronization and vertical back porch lines minus 1.
The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

46.8.5 LTDC active width configuration register (LTDC_AWCR)

Address offset: 0x010

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (HSYNC width + HBP + active width - 1), and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (VSYNC height + BVBP + active height - 1). Refer to *Figure 393* and *Section 46.5* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AAW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AAH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AAW[11:0]**: Accumulated active width (in units of pixel clock period)
This bitfield defines the accumulated active width that includes the horizontal synchronization, horizontal back porch and active pixels minus 1. The active width is the number of pixels in active display area of the panel scan line.
Refer to device datasheet for the maximum active width supported following the maximum pixel clock.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AAH[11:0]**: Accumulated active height (in units of horizontal scan line)
This bitfield defines the accumulated height that includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel.
Refer to device datasheet for the maximum active height supported following the maximum pixel clock.

46.8.6 LTDC total width configuration register (LTDC_TWCR)

Address offset: 0x014

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (HSYNC width + HBP + active width + HFP - 1), and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (VSYNC height + BVBP + active height + VFP - 1). Refer to *Figure 393* and *Section 46.5* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TOTALH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **TOTALW[11:0]**: Total width (in units of pixel clock period)

This bitfield defines the accumulated total width that includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **TOTALH[11:0]**: Total height (in units of horizontal scan line)

This bitfield defines the accumulated height that includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1.

46.8.7 LTDC global control register (LTDC_GCR)

Address offset: 0x018

Reset value: 0x0000 2220

This register defines the LTDC global configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	SFSWTR	SFEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	DEN
rw	rw	rw	rw			w	rw					rw			rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	GAMEN	LTDCEN
	r	r	r		r	r	r		r	r	r			rw	rw

Bit 31 **HSPOL**: Horizontal synchronization polarity

This bit is set and cleared by software.

0: Horizontal synchronization polarity is active low.

1: Horizontal synchronization polarity is active high.

Bit 30 **VSPOL**: Vertical synchronization polarity

This bit is set and cleared by software.

0: Vertical synchronization is active low.

1: Vertical synchronization is active high.

Bit 29 **DEPOL**: Blanking (no data/pixel) polarity

This bit is set and cleared by software.

0: Blanking (no data/pixel) polarity is active low.

1: Blanking (no data/pixel) polarity is active high.

Bit 28 **PCPOL**: Pixel clock polarity

This bit is set and cleared by software.

0: The pixel and sync data are generated at the rising-edge of the output LCD_CLK clock.

1: The pixel and sync data are generated at the falling-edge of the output LCD_CLK clock.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **SFSWTR**: Single-frame mode software trigger
This bit is set by software and cleared by hardware.
0: No action
1: Triggers one frame

Bit 24 **SFEN**: Single-frame mode enable
This bit is set and cleared by software.
0: Single-frame disabled: a trigger (on SFSWTR) generates a continuous flow.
1: Single-frame enabled: a trigger (on SFSWTR) generates a single frame.

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CRCEN**: CRC enable
This bit is set and cleared by software.
0: CRC disabled
1: CRC enabled

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **DEN**: Dither enable
This bit is set and cleared by software.
0: Dither disabled
1: Dither enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DRW[2:0]**: Dither red width
This bitfield returns the dither red bits.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DGW[2:0]**: Dither green width
This bitfield returns the dither green bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DBW[2:0]**: Dither blue width
This bitfield returns the dither Blue bits.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GAMEN**: Gamma correction enable
This bit is set and cleared by software.
0: Gamma correction disabled (pixels bypass the gamma operator)
1: Gamma correction enabled

Bit 0 **LTDCEN**: LTDC global enable
This bit is set and cleared by software.
0: LTDC disabled
1: LTDC enabled

46.8.8 LTDC global configuration 1 register (LTDC_GC1R)

Address offset: 0x01C

Reset value: 0x6BE4 D888

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMA	CRMA	STRA	DWP	SPP	Res.	TP	LNIP	BBA	BCP	SHRA	Res.	GCT[2:0]			Res.
r	r	r	r	r		r	r	r	r	r		r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[1:0]		Res.	PRBA	WRCH[3:0]				WGCH[3:0]				WBCH[3:0]			
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **BMA**: Blind mode ability

Bit 30 **CRMA**: Configuration reading mode ability

Bit 29 **STRA**: Status register ability

Bit 28 **DWP:** Dither width programmability

Bit 27 **SPP**: Sync polarity programmability

Bit 26 Reserved, must be kept at reset value.

Bit 25 **TP**: Timing programmability

Bit 24 **LNIP**: Line-IRQ: line position programmability

Bit 23 **BBA**: Background blending ability

Bit 22 **BCP**: Background color programmability (unique color blended as background)

Bit 21 **SHRA**: Shadow register ability

Bit 20 Reserved, must be kept at reset value.

Bits 19:17 **GCT[2:0]**: Gamma correction technique implemented
000: No gamma
001: Gamma with 256 samples
010: Gamma with 8 interpolated segments
Others: Reserved

Bit 16 Reserved, must be kept at reset value.

Bits 15:14 **DT[1:0]**: Dithering technique implemented
00: No dithering
01: Ordered 4x4 Bayer
10: Ordered 4x4 custom
11: Pseudo-random LFSR

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PRBA**: Precise blending ability

Bits 11:8 **WRCH[3:0]**: Width of red channel output

Bits 7:4 **WGCH[3:0]**: Width of green channel output

Bits 3:0 **WBCH[3:0]**: Width of blue channel output

46.8.9 LTDC global configuration 2 register (LTDC_GC2R)

Address offset: 0x020

Reset value: 0x0000 BB30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOA	Res.	CRCA	SFA	SISA	ROTA	AXIIDA	OCA	EDCA	BW[2:0]			DPA	DVA	STSA	BLA
r		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **BOA:** Blending order ability
0: Blending order fixed
1: Blending order configurable

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CRCA:** CRC ability
0: CRC no computation available
1: CRC computation available

Bit 12 **SFA:** Single frame mode ability
0: Single frame not available
1: Single frame available

Bit 11 **SISA:** Second interrupt set ability
0: Second interrupt set not available
1: Second interrupt set available

Bit 10 **ROTA**: Rotation support ability

Bit 9 **AXIIDA**: AXIID ability

Bit 8 **OCA**: Output conversion ability (RGB to YCbCr)

Bit 7 **EDCA**: External display control ability

Bits 6:4 **BW[2:0]**: Bus width (log2 of number of bytes)
010: 32-bit bus
011: 64-bit bus
100: 128-bit bus
Others: Reserved

Bit 3 **DPA:** Secondary RGB output port ability

Bit 2 **DVA:** Dual-view ability

Bit 1 **STSA:** Slave timings synchronization ability

Bit 0 **BLA:** Background layer ability (pixels of background layer are read from memory)

46.8.10 LTDC shadow reload configuration register (LTDC_SRCR)

Address offset: 0x024

Reset value: 0x0000 0000

This register allows the reload, either immediately or during the vertical blanking period, of shadow register values to the active registers. Shadow registers are all layer x ones, except LTDC_L1CLUTWR and LTDC_L2CLUTWR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBR**: Vertical blanking reload request

This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).

0: No effect

1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload trigger

This bit is set by software and cleared only by hardware after reload.

0: No effect

1: The shadow registers are reloaded immediately.

46.8.11 LTDC gamma correction configuration register (LTDC_GCCR)

Address offset: 0x028

Reset value: 0x0000 0000

This register performs indirect access to the gamma correction R,G,B tables.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REN	GEN	BEN
													w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[7:0]								ADDR[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **REN**: Write trigger to the red table

0: No action

1: COMP is written at ADDR in the red table.

Bit 17 **GEN**: Write trigger to the green table

0: No action

1: COMP is written at ADDR in the green table.

Bit 16 **BEN**: Write trigger to the blue table
0: No action
1: COMP is written at ADDR in the blue table.

Bits 15:8 **COMP[7:0]**: Color component to be written, in either (or all) the R,G,B tables

Bits 7:0 **ADDR[7:0]**: Address of the R,G,B table where the COMP component is written
When LTDC_GC1R.GCT=2, the gamma is implemented with eight interpolated segment. In that case, the valid ADDR addresses are: 0, 32, 64, 96, 128, 160, 192, 224, 255.
Note: For ADDR = 255, the gamma interpolation hardware considers that the address is 256.

46.8.12 LTDC background color configuration register (LTDC_BCCR)

Address offset: 0x02C

Reset value: 0x0000 0000

This register defines the background color (RGB888).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGREEN[7:0]								BCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BCRED[7:0]**: Background color red value
This bitfield configures the background red value.

Bits 15:8 **BCGREEN[7:0]**: Background color green value
This bitfield configures the background green value.

Bits 7:0 **BCBLUE[7:0]**: Background color blue value
This bitfield configures the background blue value.

46.8.13 LTDC interrupt enable register (LTDC_IER)

Address offset: 0x034

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request, by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
								rw	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIE**: CRC error interrupt enable
This bit is set and cleared by software.
0: CRC error disabled
1: CRC error interrupt enabled

Bit 6 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: Register reload interrupt enable
This bit is set and cleared by software.
0: Register reload interrupt disabled
1: Register reload interrupt enabled

Bit 2 **TERRIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: Transfer error interrupt disabled
1: Transfer error interrupt enabled

Bit 1 **FUWIE**: FIFO underrun warning interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bit 0 **LIE**: Line interrupt enable
This bit is set and cleared by software.
0: line interrupt disabled
1: line interrupt enabled

46.8.14 LTDC interrupt status register (LTDC_ISR)

Address offset: 0x038

Reset value: 0x0000 0000

This register returns the interrupt status flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
								r	r			r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIF**: CRC error interrupt flag
0: No CRC error interrupt generated
1: CRC error interrupt generated when a bus error occurs

Bit 6 **FUIF**: FIFO underrun interrupt flag
0: No FIFO underrun interrupt generated
1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and many pixel data are read from the FIFO

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: Register reload interrupt flag
0: No register reload interrupt generated
1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: Transfer error interrupt flag
0: No transfer error interrupt generated
1: Transfer error interrupt generated when a bus error occurs

Bit 1 **FUWIF**: FIFO underrun warning interrupt flag
0: No FIFO underrun warning interrupt generated
1: FIFO underrun warning interrupt generated, if one of the layer FIFO is empty and pixel data is read from the FIFO

Bit 0 **LIF**: Line interrupt flag
0: No line interrupt generated
1: Line interrupt generated when a programmed line is reached

46.8.15 LTDC interrupt clear register (LTDC_ICR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
								w	w			w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CCRCIF**: CRC error interrupt flag clear
0: No effect
1: Clear CRCIF flag in LTDC_ISR.

Bit 6 **CFUIF**: FIFO underrun interrupt flag clear
0: No effect
1: Clear FUIF flag in LTDC_ISR.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: Register reload interrupt flag clear
0: No effect
1: Clear RRIF flag in LTDC_ISR.

Bit 2 **CTERRIF**: Transfer error interrupt flag clear
0: No effect
1: Clear TERRIF flag in LTDC_ISR.

Bit 1 **CFUWIF**: FIFO underrun warning interrupt flag clear
0: No effect
1: Clear FUWIF flag in LTDC_ISR.

Bit 0 **CLIF**: Line-interrupt flag clear
0: No effect
1: Clear LIF flag in LTDC_ISR.

46.8.16 LTDC line interrupt position configuration register (LTDC_LIPCR)

Address offset: 0x040

Reset value: 0x0000 0000

This register defines the line interrupt position. The line value to be programmed depends on timing parameters (see *Figure 393*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LIPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LIPOS[11:0]**: Line interrupt position
This bitfield configures the line interrupt position.

46.8.17 LTDC current position status register (LTDC_CPSR)

Address offset: 0x044

Reset value: 0x0000 0000

This register returns the X,Y position of the currently displayed pixel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CXPOS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CYPOS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **CXPOS[11:0]**: Current X position

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CYPOS[11:0]**: Current Y position

46.8.18 LTDC current display status register (LTDC_CDSR)

Address offset: 0x048

Reset value: 0x0000 0003

This register returns the status of the current display phase that is controlled by HSYNC, VSYNC, and horizontal/vertical data-enable signals.

Example: if the current display phase is the vertical synchronization, VSYNCS is active high. If the current display phase is the horizontal synchronization, HSYNCS is active high.

The returned status does not depend on the configured polarity in LTDC_GCR: it returns the current active display phase.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **HSYNCS**: Horizontal synchronization display status
0: Active low
1: Active high

Bit 2 **VSYNCS**: Vertical synchronization display status
0: Active low
1: Active high

Bit 1 **HDES**: Horizontal data enable display status
0: Active low
1: Active high

Bit 0 **VDES**: Vertical data enable display status
0: Active low
1: Active high

46.8.19 LTDC external display control register (LTDC_EDCR)

Address offset: 0x060

Reset value: 0x0000 0000

This register controls the external display interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	OCYCO	OCYSEL	OCYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **OCYCO**: Output conversion to YCbCr 422
This bit defines the chrominance order (whether Cb or Cr is output first).
0: Cb is output first (Y0Cb, then Y1Cr, Y2Cb and so on).
1: Cr is output first (Y0Cr, then Y1Cb, Y2Cr and so on).

Bit 26 **OCYSEL**: Output conversion to YCbCr 422
This bit selects the set of CCIR hard-wired coefficients (ITU-R BT.601 or 709).
0: Use ITU-R BT.601 set (for typically SDTV analog-like displays).
1: Use ITU-R BT.709 set (for typically HDTV digital-like displays).

Bit 25 **OCYEN**: Output conversion to YCbCr 422 enable
0: Conversion disabled
1: Conversion enabled

Bits 24:0 Reserved, must be kept at reset value.

46.8.20 LTDC interrupt enable register 2 (LTDC_IER2)

Address offset: 0x064

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request, by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
								rw	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIE**: CRC error interrupt enable
This bit is set and cleared by software.
0: CRC error disabled
1: CRC error interrupt enabled

Bit 6 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: Register reload interrupt enable
This bit is set and cleared by software.
0: Register reload interrupt disabled
1: Register reload interrupt enabled

Bit 2 **TERRIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: Transfer error interrupt disabled
1: Transfer error interrupt enabled

Bit 1 **FUWIE**: FIFO underrun warning interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bit 0 **LIE**: Line interrupt enable
This bit is set and cleared by software.
0: Line interrupt disabled
1: Line interrupt enabled

46.8.21 LTDC interrupt status register 2 (LTDC_ISR2)

Address offset: 0x068

Reset value: 0x0000 0000

This register returns the interrupt status flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
								r	r			r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIF**: CRC Error interrupt flag
0: No CRC error interrupt generated
1: CRC error interrupt generated when a bus error occurs

Bit 6 **FUIF**: FIFO underrun interrupt flag
0: No FIFO underrun interrupt generated.
1: FIFO underrun interrupt generated, if one of the layer FIFO is empty and many pixel data are read from the FIFO

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: Register reload interrupt flag
0: No register reload interrupt generated
1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: Transfer error interrupt flag
0: No transfer error interrupt generated
1: Transfer error interrupt generated when a bus error occurs

Bit 1 **FUWIF**: FIFO underrun warning interrupt flag
0: No FIFO underrun warning interrupt generated.
1: FIFO underrun warning interrupt generated, if one of the layer FIFO is empty and pixel data is read from the FIFO

Bit 0 **LIF**: Line interrupt flag
0: No line interrupt generated
1: Line interrupt generated when a programmed line is reached

46.8.22 LTDC interrupt clear register 2 (LTDC_ICR2)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
								w	w			w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CCRCIF**: Clear the CRC error interrupt flag
0: No effect
1: Clear the CRCIF flag in LTDC_ISR2.

Bit 6 **CFUIF**: Clear the FIFO underrun interrupt flag
0: No effect
1: Clear the FUIF flag in LTDC_ISR2.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: Clear register reload interrupt flag
0: No effect
1: Clear the RRIF flag in LTDC_ISR2.

Bit 2 **CTERRIF**: Clear the transfer error interrupt flag
0: No effect
1: Clear the TERRIF flag in LTDC_ISR2.

Bit 1 **CFUWIF**: Clear the FIFO underrun warning interrupt flag
0: No effect
1: Clear the FUWIF flag in LTDC_ISR2.

Bit 0 **CLIF**: Clear the line interrupt flag
0: No effect
1: Clear the LIF flag in LTDC_ISR2.

46.8.23 LTDC line interrupt position configuration register 2 (LTDC_LIPCR2)

Address offset: 0x070

Reset value: 0x0000 0000

This register defines the line interrupt position. The line value to be programmed depends on timing parameters (see *Figure 393*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LIPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LIPOS[11:0]**: Line interrupt position

46.8.24 LTDC expected CRC register (LTDC_ECRCR)

Address offset: 0x078

Reset value: 0x0000 0000

This register is used to configure the expected CRC of a frame. If the computed CRC is not as expected, an interrupt CRCIF is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECRC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ECRC[15:0]**: Expected CRC of frame

46.8.25 LTDC computed CRC register (LTDC_CCRCR)

Address offset: 0x07C

Reset value: 0x0000 0000

This register returns the computed CRC of a frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCRC[15:0]**: Computed CRC of frame

46.8.26 LTDC FIFO underrun threshold register (LTDC_FUTR)

Address offset: 0x090

Reset value: 0x0000 0010

A default value for THRE[15:0] is 0x80, that represents 128 words of 64 bits, thus 256 pixels 32bpp or 1024 luminance of a YUV420 pixel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THRE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **THRE[15:0]**: Threshold to trigger a FIFO underrun interrupt (per FIFO word, 64 bits)
Threshold above which a FIFO underrun warning becomes a FIFO underrun error.

46.8.27 LTDC layer x configuration 0 register (LTDC_LxC0R)

Address offset: 0x100 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0xFF50 A075

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARGB8888	ABGR8888	RGBA8888	BGRA8888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ARGB8888**: Pixel format, ability for argb8888

Bit 30 **ABGR8888**: Pixel format, ability for abgr8888

Bit 29 **RGBA8888**: Pixel format, ability for rgba8888

Bit 28 **BGRA888**: Pixel format, ability for bgra8888

Bit 27 **RGB565**: Pixel format, ability for rgb565

Bit 26 **BGR565**: Pixel format, ability for bgr565

Bit 25 **RGB888**: Pixel format, ability for rgb888

Bit 24 **FF**: Flexible pixel format, ability

Bit 23 **F11PC**: Blending factor 1, ability for 1.0 - (pixel_alpha * constant_alpha)

Bit 22 **F1PC**: Blending factor 1, ability for pixel_alpha * constant_alpha

Bit 21 **F11C**: Blending factor 1, ability for 1.0 - constant_alpha

Bit 20 **F1C**: Blending factor 1, ability for constant_alpha

Bit 19 **F11P**: Blending factor 1, ability for 1.0 - pixel_alpha

Bit 18 **F1P**: Blending factor 1, ability for pixel_alpha

Bit 17 **F10**: Blending factor 1,ability for 0.0

Bit 16 **F11**: Blending factor 1, ability for 1.0

Bit 15 **F21PC**: Blending factor 2, ability for 1.0 - (pixel_alpha * constant_alpha)

Bit 14 **F2PC**: Blending factor 2, ability for pixel_alpha * constant_alpha

Bit 13 **F21C**: Blending factor 2, ability for 1.0 - constant_alpha

Bit 12 **F2C**: Blending factor 2, ability for constant_alpha

Bit 11 **F21P**: Blending factor 2, ability for 1.0 - pixel_alpha

Bit 10 **F2P**: Blending factor 2, ability for pixel_alpha

Bit 9 **F20**: Blending factor 2, ability for 0.0

Bit 8 **F21**: Blending factor 2, ability for 1.0

Bit 7 **CKRA**: Color key replace ability

Bit 6 **CLUTA**: CLUT ability

Bit 5 **WINA**: Windowing ability

Bit 4 **DCP**: Default color programmability

Bit 3 **APA**: Alpha plane ability

Bit 2 **CFBPA**: Color frame buffer pitch ability

Bit 1 **CFBDA**: Color frame buffer duplication ability

Bit 0 **CKTA**: Color key transparency ability

46.8.28 LTDC layer x configuration 1 register (LTDC_LxC1R)

Address offset: 0x104 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x8000 0007, 0x8000 0007, 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSPA	YIA
													r	r	r

Bit 31 **SCA**: Scaling ability for this layer
0: Scaling not available
1: Scaling available

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **YFPA**: YCbCr 420 full-planar ability for this layer
0: Full planar not available
1: Full planar available

Bit 1 **YSPA**: YCbCr 420 semi-planar ability for this layer
0: Semi-planar not available
1: Semi-planar available

Bit 0 **YIA**: YCbCr 422 interleaved ability for this layer
0: Interleaved not available
1: Interleaved available

46.8.29 LTDC layer x reload control register (LTDC_LxRCR)

Address offset: 0x108 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
													rw	rw	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **GRMSK**: Shadow reload control, global (centralized) reload masked
This bit is set and cleared by software.
0: Global reload masked for this layer (control from LTDC_SRCR disabled)
1: Global reload active for this layer (control from LTDC_SRCR enabled)

Bit 1 **VBR**: Vertical blanking reload request
This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).
0: No effect
1: Shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload trigger
This bit is set by software and cleared only by hardware after reload.
0: No effect
1: Shadow registers are reloaded immediately.

46.8.30 LTDC layer x control register (LTDC_LxCR)

Address offset: 0x10C + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SCEN	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	Res.	Res.	CKEN	LEN
					rw	rw	rw				rw			rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SCEN**: Scaler enable
This bit is set and cleared by software.
0: Scaler disabled
1: Scaler enabled

Bit 9 **DCBEN**: Default color blending enable
This bit is set and cleared by software.
0: Blending disabled
1: Blending enabled

Bit 8 **HMEN**: Horizontal mirroring enable
This bit is set and cleared by software.
0: Mirror disabled
1: Mirror enabled (if so, the color frame buffer start address has to be set to the last byte of the first line, so for instance: if line is 100 pixels, 24 bpp, then address is set to 299)

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CLUTEN**: Color look-up table enable
This bit is set and cleared by software.
0: Color look-up table disabled
1: Color look-up table enabled
The CLUT is only meaningful for L8, AL44 and AL88 pixel format.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CKEN**: Color keying enable
This bit is set and cleared by software.
0: Color keying disabled
1: Color keying enabled: if RGB matches, then the ARGB are set to 0.

Bit 0 **LEN**: Layer enable
This bit is used to enable/disable the presence of this whole layer. it is set and cleared by software.
0: Layer disabled
1: Layer enabled

46.8.31 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)

Address offset: 0x110 + 0x100 * (x -1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the horizontal position (first and last pixel) of the window of layer 1/3.

The first visible pixel of a line is the programmed value of AHBP[11:0] + 1 in LTDC_BPCR.

The last visible pixel of a line is the programmed value of AAW[11:0] in LTDC_AWCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WHSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WHSPPOS[11:0]**: Window horizontal stop position
This bitfield configures the last visible pixel of a line of the layer window.
WHSPPOS[11:0] must be ≥ AHBP[11:0] bits + 1 (programmed in LTDC_BPCR).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WHSTPOS[11:0]**: Window horizontal start position
This bitfield configures the first visible pixel of a line of the layer window.
WHSTPOS[11:0] must be ≤ AAW[11:0] bits (programmed in LTDC_AWCR).

46.8.32 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)

Address offset: 0x114 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the vertical position (first and last line) of the layer 1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[11:0] + 1 in LTDC_BPCR.

The last visible line of a frame is the programmed value of AAH[11:0] in LTDC_AWCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WVSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WVSPPOS[11:0]**: Window vertical stop position
This bitfield configures the last visible line of the layer window.
WVSPPOS[11:0] must be ≥ AVBP[11:0] + 1 (programmed in LTDC_BPCR).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WVSTPOS[11:0]**: Window vertical start position
This bitfield configures the first visible line of the layer window.
WVSTPOS[11:0] must be ≤ AAH[11:0] (programmed in LTDC_AWCR).

46.8.33 LTDC layer x color keying configuration register (LTDC_LxCKCR)

Address offset: 0x118 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the color key value (RGB), that is used by the color keying.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKGREEN[7:0]								CKBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CKRED[7:0]**: Color key red value

Bits 15:8 **CKGREEN[7:0]**: Color key green value

Bits 7:0 **CKBLUE[7:0]**: Color key blue value

46.8.34 LTDC layer x pixel format configuration register (LTDC_LxPFCR)

Address offset: 0x11C + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB). When components are provided with less than 8 bits per component, they get expanded to 8 bits by replication of their MSB onto their LSB.

When the Alpha value is not defined in the input pixel format, Alpha is assumed for farther needs to be equal to 255 (opaque layer).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PF[2:0]**: Pixel format

This bitfield configures the pixel format.
000: ARGB8888 (32 bpp)
001: ABGR8888 (32 bpp)
010: RGBA8888 (32 bpp)
011: BGRA8888 (32 bpp)
100: RGB565 (16 bpp, A = 255)
101: BGR565 (16 bpp, A = 255)
110: RGB888 (24 bpp packed, A = 255)
111: Flexible pixel format selected (see *Section 46.8.56* and *Section 46.8.57*)

46.8.35 LTDC layer x constant alpha configuration register (LTDC_LxCACR)

Address offset: 0x120 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 00FF

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending (see LTDC_LxBFCR).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CONSTA[7:0]**: Constant alpha

This bitfield configures the constant alpha used for blending. The constant alpha is divided by 255 by hardware.
Example: if the programmed constant alpha is 0xFF, the floating alpha value is 255 / 255 = 1.

46.8.36 LTDC layer x default color configuration register (LTDC_LxDCCR)

Address offset: 0x124 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCALPHA[7:0]								DCRED[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCGREEN[7:0]								DCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DCALPHA[7:0]**: Default color alpha
This bitfield configures the default alpha value.

Bits 23:16 **DCRED[7:0]**: Default color red
This bitfield configures the default red value.

Bits 15:8 **DCGREEN[7:0]**: Default color green
This bitfield configures the default green value.

Bits 7:0 **DCBLUE[7:0]**: Default color blue
This bitfield configures the default blue value.

46.8.37 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

Address offset: 0x128 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0607, 0x0001 0607, 0x0002 0607

This register defines the blending order, and the factors BF1 and BF2. The constant alpha value is the programmed value in LTDC_LxCACR, divided by 255 by hardware.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
					rw	rw	rw						rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **BOR[2:0]**: Blending order
This bitfield selects the blending order, BOR:
000: For the most background layer (usually hidden behind others)
111: For the most foreground layer (always visible, never hidden by any other)
The BOR value is limited to the max amount of layers, minus 1. For instance, if the LTDC supports three layers (layer 1/2/3), then the BOR can be configured with only the following values: 000, 001, 010.
In case of inconsistency, like two layers sharing same BOR, the blending engine reverses to BOR[layerID] = layerID - 1, so that layer 3 gets in foreground and layer 1 is in background.
Usually, the default configuration is the following one:
BOR(layer 1) = 000
BOR(layer 2) = 001
BOR(layer 3) = 010

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **BF1[2:0]**: Default factor 1
This bitfield selects the blending factor F1.
100: Constant alpha
110: Pixel alpha x constant alpha
Others: Reserved

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **BF2[2:0]**: Blending factor 2
This bitfield selects the blending factor F2.
101: 1 - constant alpha
111: 1 - (pixel alpha x constant alpha)
Others: Reserved

46.8.38 LTDC layer x burst length configuration register (LTDC_LxBLCR)

Address offset: 0x12C + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the length of the bursts requested by layer x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **BL[4:0]**: Burst length
0x00: maximum burst length (16 words 64 bits, thus 128 Bytes)
0x01: 1 word (of 64 bits) per burst
0x10: 16 words (of 64 bits) per burst
Others: Reserved

46.8.39 LTDC layer x planar configuration register (LTDC_LxPCR)

Address offset: 0x130 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines, for layer x, the configuration when accessing planar or semi-planar buffers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM[1:0]		YCEN	Res.	Res.	Res.
						rw	rw	rw	rw	rw	rw	rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **YREN**: Y rescale enable for the color dynamic range
When enabled, incoming Y values in range 16 to 235, are re-scaled to range 0 to 255.
0: Rescaling disabled (input component thus assumed provided in 0 to 255)
1: Rescaling enabled (input component thus assumed provided in 16 to 235).

Bit 8 **OF**: Odd pixel first
This bit defines if the byte 0 of a word (in LSB) contains the odd pixel.
0: Odd pixel disabled (thus even pixel on byte 0)
1: Odd pixel enabled (thus odd pixel on byte 0)

Bit 7 **CBF**: Cb component first
This bit defines if the byte 0 and 1 of a word (in LSB) contains the Cb component. The setting impacts only for interleaved and semi-planar modes, as it has no meaning in full-planar mode.
0: Cb disabled (thus Cr component is on byte 0 and 1)
1: Cb enabled (thus Cb component is on byte 0 and 1)

Bit 6 **YF**: Y component first
This bit defines if the byte 0 of a word (in LSB) contains the Y component.
0: Y component disabled (thus Cr or Cb component is on byte 0)
1: Y component enabled (thus Y component is on byte 0)

Bits 5:4 **YCM[1:0]**: YCbCr conversion mode
This bitfield defines the type of input that is considered and converted to a YCbCr 444.
00: Interleaved 422 (Cb and Cr component are replicated horizontally for pixels P and P+1)
01: Semi-Planar 420: (Cb and Cr component are replicated horizontally and vertically.The layer main configuration defines the access to the Y buffer, and auxiliary registers define the access to the Cb and Cr buffers).
10: Full-Planar 420: (Cb and Cr component are replicated horizontally and vertically. The layer main configuration defines the access to the Y buffer, and auxiliary registers define the access to the Cb and Cr buffers).
11: Reserved

Bit 3 **YCEN**: YCbCr-to-RGB conversion enable
0: Conversion disabled
1: YCbCr conversion enabled, using the YCM setting above

Bits 2:0 Reserved, must be kept at reset value.

46.8.40 LTDC layer x color frame buffer address register (LTDC_LxCFBAR)

Address offset: 0x134 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the color frame buffer start address, which has to point to the address where the pixel data of the top-left pixel of a layer is stored in the frame buffer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFBADD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFBADD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFBADD[31:0]**: Color frame buffer start address
This bitfield defines the color frame buffer start address.

46.8.41 LTDC layer x color frame buffer length register (LTDC_LxCFBLR)

Address offset: 0x138 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the color frame buffer line length and pitch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CFBP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CFBLL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value. Upon reading, this bit returns the value of the bit 30.

Bits 30:16 **CFBP[14:0]**: Color frame buffer pitch in bytes
This bitfield defines the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.
Negative values (with MSB bit = 1) are allowed, to read the buffer from bottom to top, and thus to flip is vertically. When vertically flipped, as the address register must provide the address of the first line to be read, the address register must point to the start of the bottom line of the buffer.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CFBLL[13:0]**: Color frame buffer line length
This bitfield defines the length of one line of pixels in bytes + 7.
The line length is computed as follows: active high width * number of bytes per pixel + 7.

46.8.42 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)

Address offset: 0x13C + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the number of lines in the color frame buffer. The number of lines and line-length settings define how much data are fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled. The start address and pitch settings define the correct start of every line in memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CFBLNBR[11:0]**: Frame buffer line number

This bitfield defines the number of lines in the frame buffer that corresponds to the active high width.

46.8.43 LTDC layer x auxiliary frame buffer address 0 register (LTDC_LxAFBA0R)

Address offset: 0x140 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the start address for the auxiliary 0 frame buffer. It has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer. The auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats: the color frame buffer is used to store the Y components, while the auxiliary buffers 0 and 1 are used to store the other one (for semi-planar) or two (for full-planar) planes of components.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFBADD0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFBADD0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFBADD0[31:0]**: Frame buffer start address

46.8.44 LTDC layer x auxiliary frame buffer address 1 register (LTDC_LxAFBA1R)

Address offset: 0x144 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the start address for the auxiliary 0 frame buffer. It has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer (the auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFBADD1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFBADD1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFBADD1[31:0]**: Auxiliary frame buffer start address

46.8.45 LTDC layer x auxiliary frame buffer length register (LTDC_LxAFBLR)

Address offset: 0x148 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the auxiliary 0 and 1 frame buffer line length and pitch (the auxiliary0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AFBP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AFBLL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value. Upon reading, this bit returns the value of the bit 30.

Bits 30:16 **AFBP[14:0]**: Auxiliary frame buffer pitch in bytes

This bitfield defines the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.
Negative values (with MSB bit = 1) are allowed, to read the buffer from bottom to top, and thus to flip is vertically. When vertically flipped, as the address register must provide the address of the first line to be read, the address register must point to the start of the bottom line of the buffer.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **AFBLL[13:0]**: Auxiliary frame buffer line length

This bitfield defines the length of one line of pixels in bytes + 7.
The line length is computed as follows: active high width * number of bytes per pixel + 7.

46.8.46 LTDC layer x auxiliary frame buffer line number register (LTDC_LxAFBLNR)

Address offset: 0x14C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the number of lines in the auxiliary 0 and 1 frame buffers (the auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats). The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled. The start address and pitch settings define the correct start of every line in memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AFBLNBR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **AFBLNBR[11:0]**: Auxiliary frame buffer line number
This bitfield defines the number of lines in the frame buffer that corresponds to the active high width.

46.8.47 LTDC layer x CLUT write register (LTDC_LxCLUTWR)

Address offset: 0x150 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

This register defines the CLUT address and the RGB value. The CLUT write register must only be configured during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in LTDC_LxCR. The CLUT is only meaningful for L8, AL44, and AL88 pixel format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **CLUTADD[7:0]**: CLUT address
This bitfield configures the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: Red value
This bitfield configures the red value.

Bits 15:8 **GREEN[7:0]**: Green value
This bitfield configures the green value.

Bits 7:0 **BLUE[7:0]**: Blue value
This bitfield configures the blue value.

46.8.48 LTDC layer x scaler input size register (LTDC_LxSISR)

Address offset: 0x154 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler input size (width and height).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SIV[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SIH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SIV[11:0]**: Scaler vertical input size (input height)

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SIH[11:0]**: Scaler horizontal input size (input width)

46.8.49 LTDC layer x scaler output size register (LTDC_LxSOSR)

Address offset: 0x158 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler output size (width and height).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SOV[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SOH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SOV[11:0]**: Scaler vertical output size (output height)

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SOH[11:0]**: Scaler horizontal output size (output width)

46.8.50 LTDC layer x scaler vertical scaling factor register (LTDC_LxSVSFR)

Address offset: 0x15C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler vertical scaling factor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVF[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SVF[15:0]**: Scaler vertical scaling factor

46.8.51 LTDC layer x scaler vertical scaling phase register (LTDC_LxSVSPR)

Address offset: 0x160 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler vertical scaling phase.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SVP[15:0]**: Scaler vertical scaling factor

46.8.52 LTDC layer x scaler horizontal scaling factor register (LTDC_LxSHSFR)

Address offset: 0x164 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler horizontal scaling factor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHF[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SHF[15:0]**: Scaler horizontal scaling factor

46.8.53 LTDC layer x scaler horizontal scaling phase register (LTDC_LxSHSPR)

Address offset: 0x168 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the scaler horizontal scaling phase.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SHP[15:0]**: Scaler horizontal scaling phase

46.8.54 LTDC layer x conversion YCbCr RGB 0 register (LTDC_LxCYR0R)

Address offset: 0x16C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the coefficients to convert from YCbCr to RGB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **CB2B[9:0]**: Cb-to-Blue coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **CR2R[9:0]**: Cr-to-Red coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

46.8.55 LTDC layer x conversion YCbCr RGB 1 register (LTDC_LxCYR1R)

Address offset: 0x170 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the coefficients to convert from YCbCr to RGB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **CB2G[9:0]**: Cb-to-Green coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **CR2G[9:0]**: Cr-to-Green coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

46.8.56 LTDC layer x flexible pixel format 0 register (LTDC_LxFPF0R)

Address offset: 0x174 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0002 1100

This register is one of the registers that define the flexible color format, by specifying the location and width of all A, R, G, B components in a memory word. Components A and R are defined in this specific register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLEN[1:0]		RPOS[4:0]					ALEN[3:0]				APOS[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **RLEN[3:0]**: Width of the red component (in bits)

Bits 13:9 **RPOS[4:0]**: Location of the red component inside the pixel memory word (in bits)

Bits 8:5 **ALEN[3:0]**: Width of the alpha component (in bits)

Bits 4:0 **APOS[4:0]**: Location of the alpha component inside the pixel memory word (in bits)

46.8.57 LTDC layer x flexible pixel format 1 register (LTDC_LxFPF1R)

Address offset: 0x178 + 0x100 * (x - 1), (x = 1 to 3)

Reset value: 0x0012 3110

This register is one of the registers that define the flexible color format, by specifying the location and width of all A, R, G, B components in a memory word. Components G and B are defined in this specific register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE[2:0]			BLEN[3:2]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLEN[1:0]		BPOS[4:0]					GLEN[3:0]				GPOS[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **PSIZE[2:0]**: Pixel size (in bytes)

Bits 17:14 **BLEN[3:0]**: Width of the blue component (in bits)

Bits 13:9 **BPOS[4:0]**: Location of the blue component inside the pixel memory word (in bits)

Bits 8:5 **GLEN[3:0]**: Width of the green component (in bits)

Bits 4:0 **GPOS[4:0]**: Location of the green component inside the pixel memory word (in bits)

46.8.58 LTDC register map

Table 431. LTDC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LTDC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[7:0]								MINVER[7:0]								REV[7:0]							
	Reset value									0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x004	LTDC_LCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNBR7:0]							
	Reset value																									0	0	0	0	0	0	1	1
	Reset value																									0	0	0	0	0	0	1	0
0x008	LTDC_SSCR	Res.	Res.	Res.	Res.	Res.	HSW[11:0]											Res.	Res.	Res.	Res.	VSH[11:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x00C	LTDC_BPCR	Res.	Res.	Res.	Res.	AHBP[11:0]												Res.	Res.	Res.	Res.	AVBP[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010	LTDC_AWCR	Res.	Res.	Res.	Res.	AAW[11:0]												Res.	Res.	Res.	Res.	AAH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x014	LTDC_TWCR	Res.	Res.	Res.	Res.	TOTALW[11:0]												Res.	Res.	Res.	Res.	TOTALH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Res.	SFEXEN	SFSWTR	SFEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	DEN	Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	ROTEN	GAMEN	LTDCEN	
	Reset value	0	0	0	0		0	0	0					0			0		0	1	0		0	1	0		0	1	0		0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	LTDC_GC1R	BMA	CRMA	STRA	DWP	SPP	Res.	TP	LNIP	BBA	BCP	SHRA	Res.	GCT[2:0]			Res.	DT [1:0]		Res.	PRBA	WRCH[3:0]				WGCH[3:0]				WBCH[3:0]			
	Reset value	0	1	1	0	1		1	1	1	1	1		0	1	0		1	1		1	1	0	0	0	1	0	0	0	1	0	0	0
0x020	LTDC_GC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOA	Res.	CRCA	SFA	SISA	ROTA	AXIIDA	OCA	EDCA	BW [2:0]			DPA	DVA	STSA	BLA
	Reset value	0x0000 BB30																															
0x024	LTDC_SRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
	Reset value																															0	0
0x028	LTDC_GCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REN	GEN	BEN	COMP[7:0]								ADDR[7:0]							
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	LTDC_BCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]								BCGREEN[7:0]								BCBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Reserved																															
0x034	LTDC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
	Reset value																									0	0			0	0	0	0
0x038	LTDC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
	Reset value																									0	0			0	0	0	0
0x03C	LTDC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
	Reset value																									0	0			0	0	0	0
0x040	LTDC_LIPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x044	LTDC_CPSR	Res.	Res.	Res.	Res.	CXPOS[11:0]												Res.	Res.	Res.	Res.	CYPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x048	LTDC_CDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
	Reset value																													0	0	1	1
0x04C-0x05C	Reserved	Reserved																															
0x060	LTDC_EDCR	Res.	Res.	Res.	Res.	OCYCO	OCYSEL	OCYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0																									
0x064	LTDC_IE2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
	Reset value																									0	0			0	0	0	0
0x068	LTDC_ISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
	Reset value																									0	0			0	0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x06C	LTDC_ICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
	Reset value																									0	0			0	0	0	0
0x070	LTDC_LIPCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Reserved																															
0x078	LTDC_ECRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	LTDC_CCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080-0x08C	Reserved	Reserved																															
0x090	LTDC_FUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved	Reserved																															
0x100	LTDC_L1C0R	ARGB8888	ABGR8888	RGBA8888	BGRA8888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11	F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
	Reset value	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1
0x104	LTDC_L1C1R	SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSP	YIA
	Reset value	1																													1	1	1
0x108	LTDC_L1RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
	Reset value																														1	0	0
0x10C	LTDC_L1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCEN	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	HPDEN	VPDEN	CKEN	LEN
	Reset value																						0	0	0				0	0	0	0	0
0x110	LTDC_L1WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	Res.	WHSTPOS[11:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x114	LTDC_L1WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x118	LTDC_L1CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C	LTDC_L1PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x120	LTDC_L1CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x124	LTDC_L1DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x128	LTDC_L1BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR[2:0]			Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value														0	0	0						1	1	0						1	1	1
0x12C	LTDC_L1BLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
	Reset value																												0	0	0	0	0
0x130	LTDC_L1PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM [1:0]		YCEN	Res.	Res.	Res.
	Reset value																							0	0	0	0	0	0	0			
0x134	LTDC_L1CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138	LTDC_L1CFBLR	Res.	CFBP[14:0]															Res.	Res.	CFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x13C	LTDC_L1CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x140	LTDC_L1AFBA0R	AFBADD0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	LTDC_L1AFBA1R	AFBADD1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148	LTDC_L1AFBLR	Res.	AFBP[14:0]															Res.	Res.	AFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14C	LTDC_L1AFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x150	LTDC_ L1CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	LTDC_L1SISR	Res.	Res.	Res.	Res.	SIV[11:0]												Res.	Res.	Res.	Res.	SIH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x158	LTDC_L1SOSR	Res.	Res.	Res.	Res.	SOV[11:0]												Res.	Res.	Res.	Res.	SOH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x15C	LTDC_L1SVSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	LTDC_L1SVSPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164	LTDC_L1SHSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	LTDC_L1SHSPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x16C	LTDC_L1CYR0R	Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x170	LTDC_L1CYR1R	Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x174	LTDC_L1FPF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:0]				RPOS[4:0]					ALEN[3:0]				APOS[4:0]				
	Reset value															1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x178	LTDC_L1FPF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE [2:0]			BLEN[3:0]				BPOS[4:0]					GLEN[3:0]				GPOS[4:0]				
	Reset value												1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0
0x17C-0x1FC	Reserved	Reserved																															
0x200	LTDC_L2C0R	ARGB8888	ABGR8888	RGBA8888	BGRA8888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11	F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
	Reset value	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1
0x204	LTDC_L2C1R	SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSP	YIA
	Reset value	1																													1	1	1
0x208	LTDC_L2RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
	Reset value																														1	0	0
0x20C	LTDC_L2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCEN	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	HPDEN	VPDEN	CKEN	LEN
	Reset value																						0	0	0				0	0	0	0	0
0x210	LTDC_L2WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	Res.	WHSTPOS[11:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x214	LTDC_L2WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x218	LTDC_L2CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x21C	LTDC_L2PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x220	LTDC_L2CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x224	LTDC_L2DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x228	LTDC_L2BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR[2:0]			Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value														0	0	0						1	1	0						1	1	1
0x22C	LTDC_L2BLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
	Reset value																												0	0	0	0	0
0x230	LTDC_L2PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM[1:0]		YCEN	Res.	Res.	Res.
	Reset value																							0	0	0	0	0	0	0			
0x234	LTDC_L2CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x238	LTDC_L2CFBLR	Res.	CFBP[14:0]															Res.	Res.	CFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x23C	LTDC_L2CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]												Res.
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	
0x240	LTDC_L2AFBA0R	AFBADD0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x244	LTDC_L2AFBA1R	AFBADD1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x248	LTDC_L2AFBLR	Res.	AFBP[14:0]															Res.	Res.	AFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24C	LTDC_L2AFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x250	LTDC_ L2CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x254	LTDC_L2SISR	Res.	Res.	Res.	Res.	SIV[11:0]												Res.	Res.	Res.	Res.	SIH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x258	LTDC_L2SOSR	Res.	Res.	Res.	Res.	SOV[11:0]												Res.	Res.	Res.	Res.	SOH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x25C	LTDC_L2SVSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x260	LTDC_L2SVSPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x264	LTDC_L2SHSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x268	LTDC_L2SHSPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x26C	LTDC_L2CYR0R	Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x270	LTDC_L2CYR1R	Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x274	LTDC_L2FPF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:0]				RPOS[4:0]					ALEN[3:0]				APOS[4:0]				
	Reset value															1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x278	LTDC_L2FPF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE [2:0]			BLEN[3:0]				BPOS[4:0]					GLEN[3:0]				GPOS[4:0]				
	Reset value												1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0
0x27C-0x2FC	Reserved	Reserved																															
0x300	LTDC_L3C0R	ARGB8888	ABGR8888	RGBA8888	BGRA888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11	F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
	Reset value	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1
0x304	LTDC_L3C1R	SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSP	YIA
	Reset value	0																													0	0	0

Table 431. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	LTDC_L3RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
	Reset value																														1	0	0
0x30C	LTDC_L3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCEN	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	HPDEN	VPDEN	CKEN	LEN
	Reset value																						0	0	0				0	0	0	0	0
0x310	LTDC_L3WHPCR	WHSPPOS[15:0]																WHSTPOS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	LTDC_L3WVPCR	WVSPPOS[15:0]																WVSTPOS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	LTDC_L3CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	LTDC_L3PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x320	LTDC_L3CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x324	LTDC_L3DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x328	LTDC_L3BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR[2:0]			Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value														0	0	0						1	1	0						1	1	1
0x32C	LTDC_L3BLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
	Reset value																												0	0	0	0	0
0x330	LTDC_L3PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM [1:0]		YCEN	Res.	Res.	Res.
	Reset value																							0	0	0	0	0	0	0			
0x334	LTDC_L3CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x338	LTDC_L3CFBLR	Res.	CFBP[14:0]															Res.	Res.	CFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x33C	LTDC_L3CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x340-0x34C	Reserved	Reserved																															
0x350	LTDC_ L3CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x354-0x370	Reserved	Reserved																															
0x374	LTDC_L3FPF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:0]				RPOS[4:0]						ALEN[3:0]				APOS[4:0]			
	Reset value															1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
0x378	LTDC_L3FPF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE [2:0]			BLEN[3:0]				BPOS[4:0]						GLEN[3:0]				GPOS[4:0]			
	Reset value												1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

47 True random number generator (RNG)

47.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is an NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

47.2 RNG main features

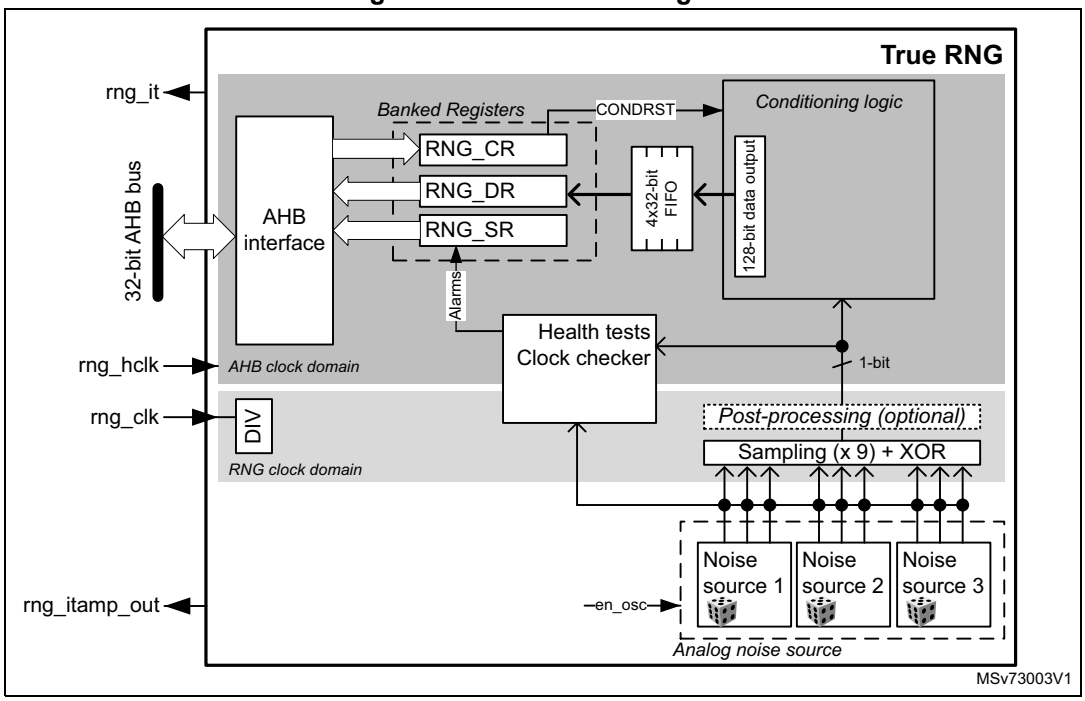
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the NIST configuration, produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

47.3 RNG functional description

47.3.1 RNG block diagram

Figure 394 shows the RNG block diagram.

Figure 394. RNG block diagram



47.3.2 RNG internal signals

Table 432 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 432. RNG internal input/output signals

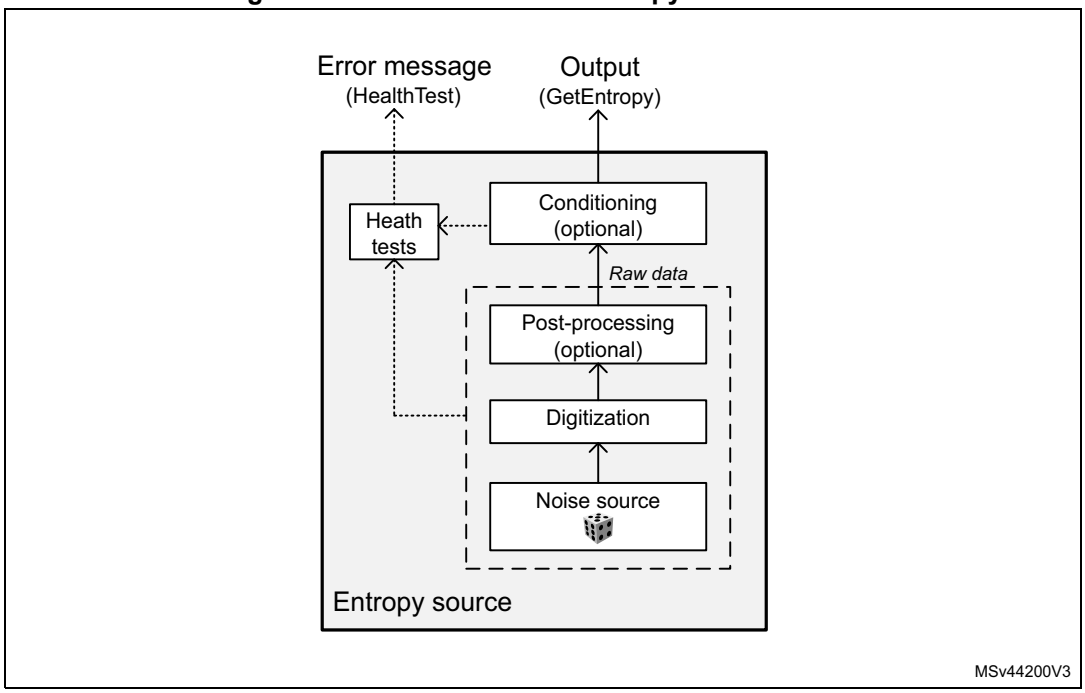
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

47.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, it integrates all the required NIST components shown in *Figure 395*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 395. NIST SP800-90B entropy source model



Noise source

This component contains the nondeterministic, entropy-providing activity, ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 47.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 434*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with the possibility, for the software, to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 47.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 47.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 47.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 47.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCRx (configuration A in *Section 47.6.2*). The description below refers to the RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:

- Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 47.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 47.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCR register. See Section 47.6 for details.

47.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 396*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 47.5*.

Figure 396. RNG initialization overview

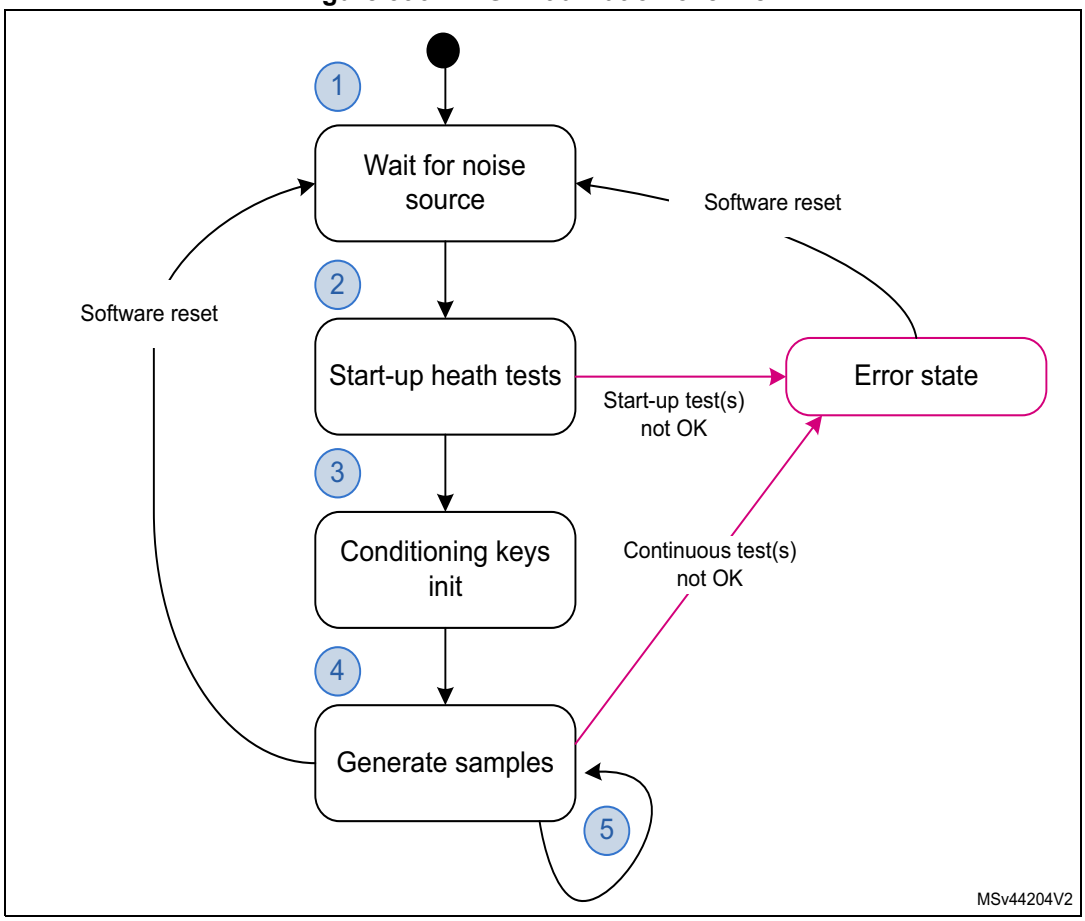


Figure 396 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers. Software reset with CONFIGLOCK set preserves the configuration.

47.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check in *Section 47.6* if a specific RNG configuration is required for the application.
   – If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.

- If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 47.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check in *Section 47.6* if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If the above conditions are true, read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, use the recovery sequence described in *Section 47.3.7*.

Note: When data are not ready (DRDY = 0) RNG_DR returns 0. It is recommended to verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the configuration B, described in *Section 47.6*. The gain in random number generation speed is summarized in *Section 47.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 47.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 47.3.3*.

47.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 47.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 47.3.1* for details (AHB and RNG clock domains).

47.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 47.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 47.3.5*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 47.3.5*.

When ARDIS is cleared (auto-reset enabled), clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.

When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the autoreset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

47.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 47.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate the RNG kernel clock, the software must issue the following sequence: first, disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR, and finally, disable the RNG kernel clock using the RCC.

Before entering a power mode where RNG is deactivated (like Stop mode), the software must first disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

47.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 47.3.7*
- Clock error, see *Section 47.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 433*.

Table 433. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 47.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

47.5 RNG processing time

In the recommended configuration A or C described in *Table 434*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

If f_{RNG}= 48 MHz, $f_{threshold}$ = 1.6 x f_{RNG} = 77 MHz. The value of N is defined in *Section 47.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected, the performance figure becomes:

- 206 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold}$ = 6.5 x f_{RNG}.

47.6 RNG entropy source validation

47.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

47.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk = 64 MHz, with CED bit cleared in RNG_CR.
- Configurations are described in *Table 434* and *Table 435*. Only configuration A can be certified NIST SP800-90B. Refer to *Table 436* to select the best configuration for the application.
- Configuration C can be used when configuration A is not present in AN4230, or is not flagged as certified.

Table 434. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_HTCR0 register	RNG_NSCR register
	NISTC bit	RNG_CONFIG1 [7:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0]	RNG_CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x01	0x1(2)	0x1	0x1	1	0x0000 AAC7(3)	Default
C	0	0x82		0x1	0xF	2		Default

1. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N = 1, 0x1 to N = 2, 0x2 to N = 3, and 0x3 to N = 4.
2. The noise source sampling must be 32 MHz or less. If the RNG clock is different from 64 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 47.7.1* for details.
3. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

Table 435. Additional health test configurations

Configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	default	default	default
B and C	default	default	default

Table 436. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, an NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on *www.st.com*.

47.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

47.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

0: Writes to the RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 47.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 47.6* for proposed configuration.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 47.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable
This bit enables/disables the true random number generator.
0: True random number generator disabled.
1: True random number generator enabled.
Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.
When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

Bits 1:0 Reserved, must be kept at reset value.

47.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status
This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.
0: No faulty sequence detected
1: At least one faulty sequence is detected. See SECS bit description for details.
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status
This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.
0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)
1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy
This flag indicates whether RNG is idle or busy.
0: Idle
1: Busy
RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.
More information on BUSY usage can also be found in *Section 47.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

47.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

47.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

47.7.5 RNG health test control register x (RNG_HTCRx)

Address offset: 0x010 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 47.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

47.7.6 RNG hardware configuration register (RNG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0303

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								CFG1[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 Reserved, must be kept at reset value.

Bits 31:16 **CFG3[7:0]**: HW Generic 3
0x0: Analog noise source
0x1: Analog or digital noise source

Bits 15:8 **CFG2[7:0]**: Hardware generic 2
This field returns the number of rng buses (rngbus_nb).

Bits 7:0 **CFG1[7:0]**: Hardware generic 1
This field returns the triple_oscillator_nb generic value (0x3).

47.7.7 RNG version register (RNG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: RNG major revision

Bits 3:0 **MINREV[3:0]**: RNG minor revision

47.7.8 RNG identification register (RNG_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RNG identification code

47.7.9 RNG size identification register (RNG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RNG size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1 Kbyte address decoding)

47.7.10 RNG register map

Table 437. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 437. RNG register map and reset map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3 [2:0]			EN_OSC2 [2:0]			EN_OSC1 [2:0]		
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3EC 0x20C	Reserved	Reserved																															
0x3F0	RNG_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[7:0]								CFG2[7:0]								CFG1[7:0]							
	Reset value									0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
0x3F4	RNG_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	1	0	0	0	1	0	0
0x3F8	RNG_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	RNG_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

48 Cryptographic processor (SAES)

48.1 SAES introduction

The cryptographic processor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP, and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128, 192, or 256 bits.

The key wrap feature is also available, decrypting AES keys to use in SAES or in the faster CRYP1 or CRYP2 peripheral.

SAES can load, by hardware, STM32 hardware secret keys (boot hardware key BHK, and derived hardware unique key DHUK), usable but not readable by the application.

SAES supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

SAES is hardware-linked with the true random number generator (TRNG) and with the CRYP1 and CRYP2 peripherals.

48.2 SAES main features

- Compliance with NIST *"Advanced encryption standard (AES)*, *FIPS publication 197*" from November 2001
- Multiple AES symmetric key block cipher operating modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 16-byte data block processing, supporting cipher key lengths of 128, 192 and 256 bits
  - 82 or 102 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus; under SAES control, optional key sharing with the faster CRYP1 or CRYP2 peripheral
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- Eight 32-bit write-only registers (256 bits of information) for storing cryptographic keys
- Optional hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
- Security context enforcement for keys
- Four 32-bit registers for storing initialization vectors
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB target peripheral, accessible only via 32-bit single-word write accesses (other write accesses generate bus errors and are ignored); no restrictions on reads
- Ability to suspend a message by software (in CPU mode) and later resume it, except in GCM/GMAC modes. This allows SAES to process a higher-priority message.

48.3 SAES implementation

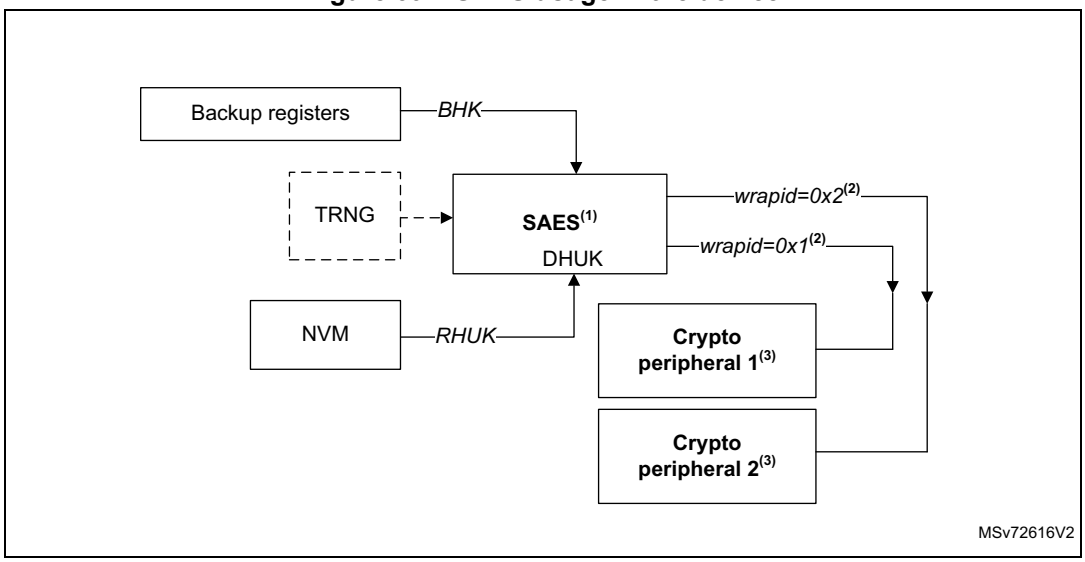
The devices have one SAES peripheral, implemented as per the following table. For comparison, the SAES key share target CRYP1 and CRYP2 peripherals are also presented.

Table 438. CRYP1 versus SAES features

Modes or features(1)	CRYP1 and CRYP2	SAES
ECB, CBC chaining	X	X
CTR, CCM, and GCM chaining	X	X
AES 128-bit ECB encryption latency in cycles	14	82
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and CRYP1 or CRYP2	X	X
Key sizes in bits	128, 192, 256	128, 192, 256

1. X = supported.

Figure 397. SAES usage in the device



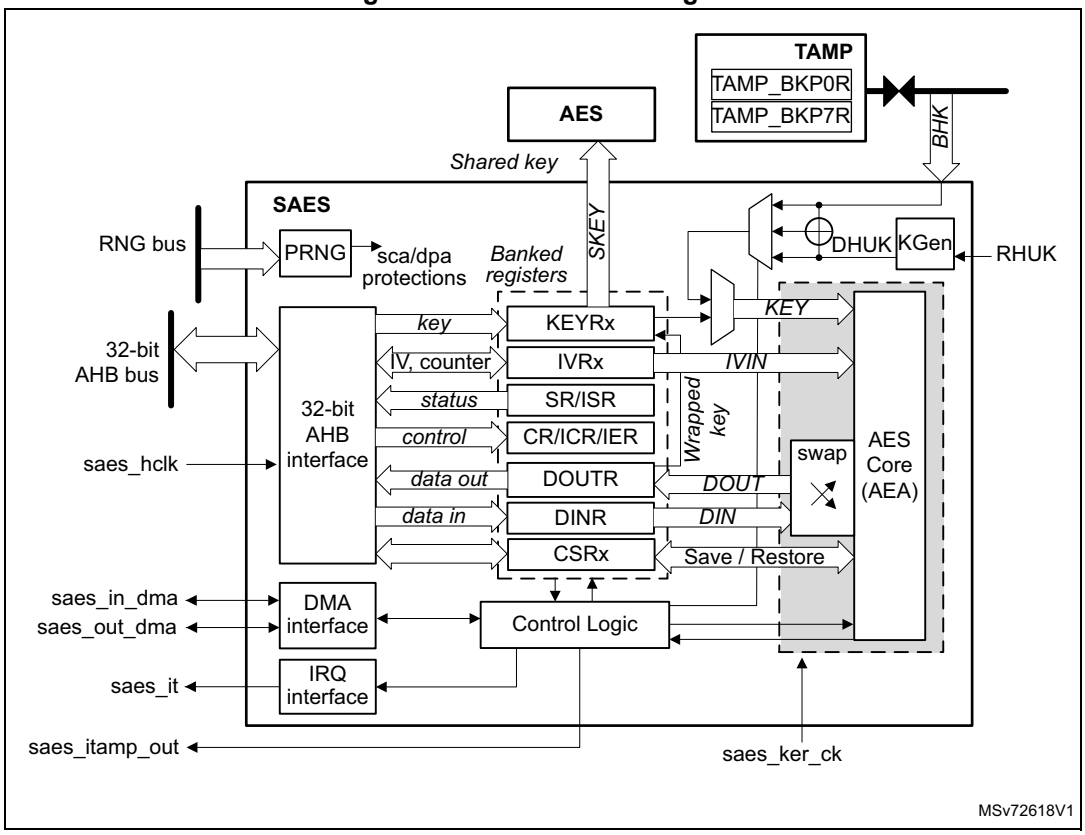
1. Protected against side-channel attack.
2. Configuration bits when WRAPEN is set.
3. Refer to *Table 438* for information on instances of crypto peripherals on the device and their names.

48.4 SAES functional description

48.4.1 SAES block diagram

Figure 398 shows the SAES block diagram.

Figure 398. SAES block diagram



1. AES represents the CRYP1 or CRYP2 peripheral.

48.4.2 SAES internal signals

Table 439 describes the user relevant internal signals interfacing the SAES peripheral.

Table 439. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge

Table 439. SAES internal input/output signals

Signal name	Signal type	Description
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required before SAES can be used again.
RHUK	Input	256-bit root hardware unique key (nonvolatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant backup registers and written during boot. Once written, this key cannot be read nor written by any application until the next product reset.

1. Connected to a set of backup registers in the TAMP peripheral that are written, then read/write locked, by the application software (see *Section 48.4.17* for details).

48.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It also has a dedicated kernel clock controlled through the RCC.

The SAES has a dedicated reset bit in the RCC.

48.4.4 SAES symmetric cipher implementation

The cryptographic processor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 440*. Those functions can be certified NIST PUB 140-3.

Table 440. SAES approved AES symmetric key functions

Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 192, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

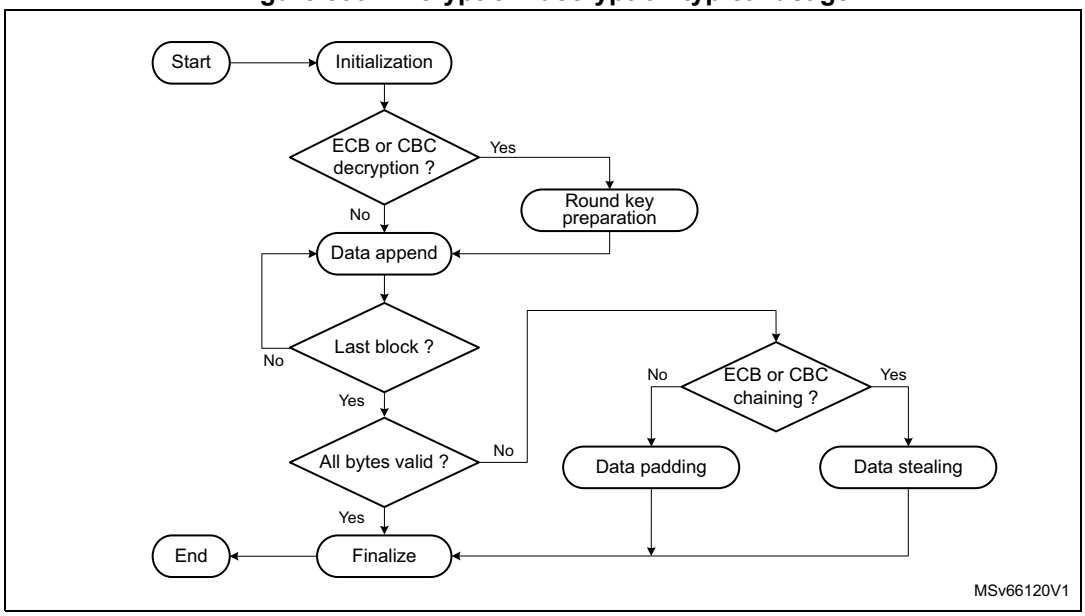
SAES can be used directly by the CPU, or indirectly using two DMA channels (one for the plaintext, one for the ciphertext).

When the CPU is used, it is possible to suspend then resume any processing by SAES (except in GCM and GMAC operating modes), following the sequence described in *Section 48.4.8*.

48.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 399. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the operating (chaining) mode. Refer to the sections describing the operating modes.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 48.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

General conditions for data processing

For information on how to suspend a message processing to process a higher priority message, refer to *Section 48.4.8: SAES suspend and resume operations*.

Appending data via CPU polling

This method uses the CCF flag polling to control data appending through the following sequence:

1. When KEYVALID is set, enable the SAES peripheral, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following subsequence until the payload is entirely processed:
   a) Write one block of data (four words) into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read one block of data (four words) from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of nonvalid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: The NPBLB[3:0] bitfield is not used in a header phase of GCM, GMAC, and CCM chaining modes.

Appending data via CPU in interrupt mode

The method uses an interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. When KEYVALID is set, enable the SAES peripheral, by setting EN (if not already done).
3. Write the first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write the next four input data words into the SAES_DINR register and quit the interrupt service routine.

SAES tolerates delays between consecutive read or write operations, allowing, for example, the system to serve an interrupt from another peripheral between two SAES computations.

Note: The NPBLB[3:0] bitfield is not used during the header phase of the GCM, GMAC, and CCM chaining modes.

Appending data via DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 48.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data via*

CPU polling must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. When KEYVALID is set, enable the SAES peripheral, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the outgoing transfer completion, get the SAES-processed data from the memory.

When this method is used, suspend/resume operations are not supported (see *Section 48.4.8*).

The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

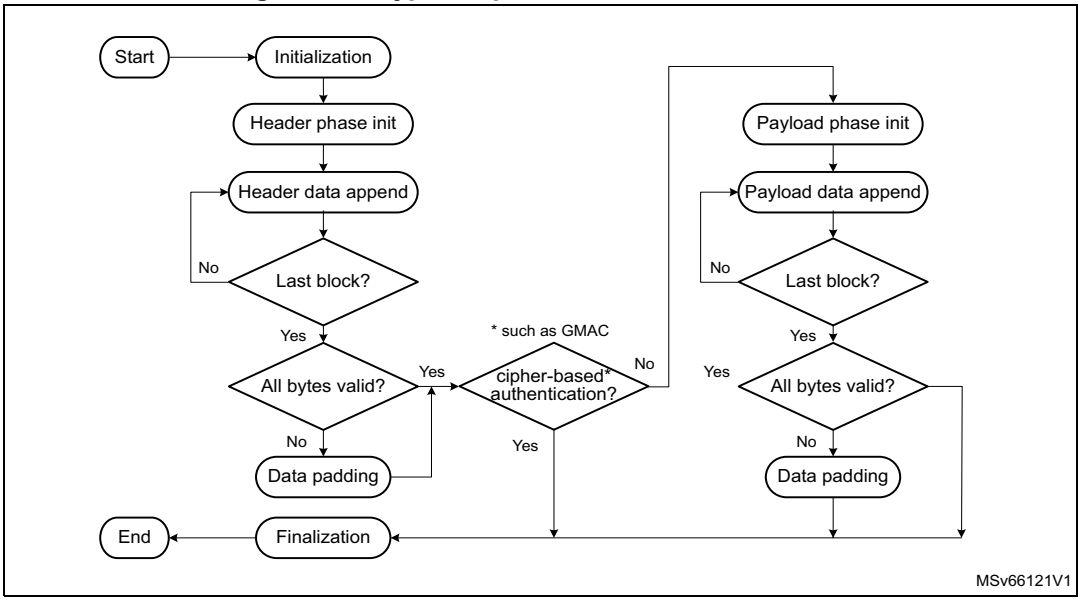
Aborting data appending process

To abort the process of appending data, set then clear the IPRST bit of the SAES_CR register.

48.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 400. Typical operation with authentication



Section 48.4.11: SAES Galois/counter mode (GCM) and *Section 48.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the above figure. The detailed sequence supported by SAES is described in *Section 48.4.12: SAES Galois message authentication code (GMAC)*.

48.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

For modes other than ECB or CBC, since SAES does not implement an automatic data padding operation for the last block, the application must follow the recommendation provided in *Section 48.4.5* to manage messages whose sizes are not multiples of 16 bytes.

48.4.8 SAES suspend and resume operations

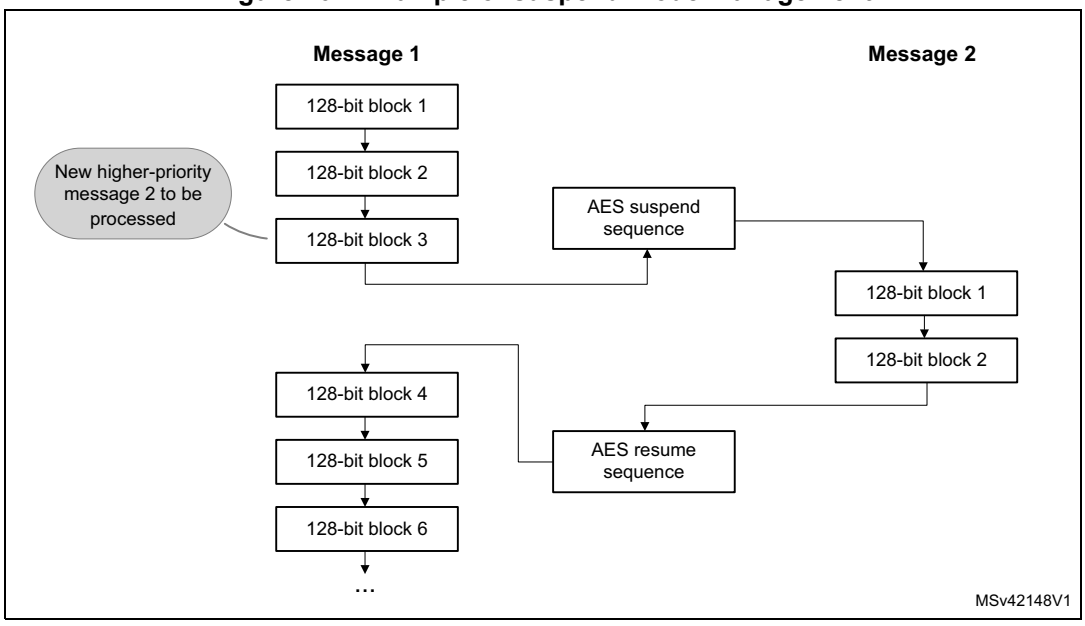
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

Note: Suspend and resume operations are only supported when SAES is used in CPU mode.

Figure 401 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2. It is recommended to set the IPRST bit of the SAES_CR register before transferring control of SAES to a potentially less secure application.

Figure 401. Example of suspend mode management



A detailed description of suspend and resume operations is provided in the sections dedicated to each chaining mode.

48.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 402 and *Figure 403* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[3:0] with 0x0.

Figure 402. ECB encryption

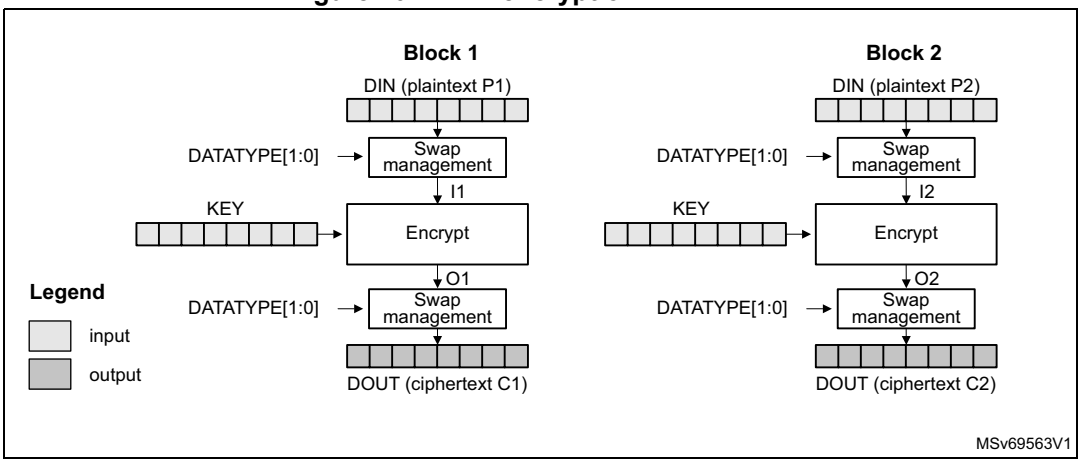
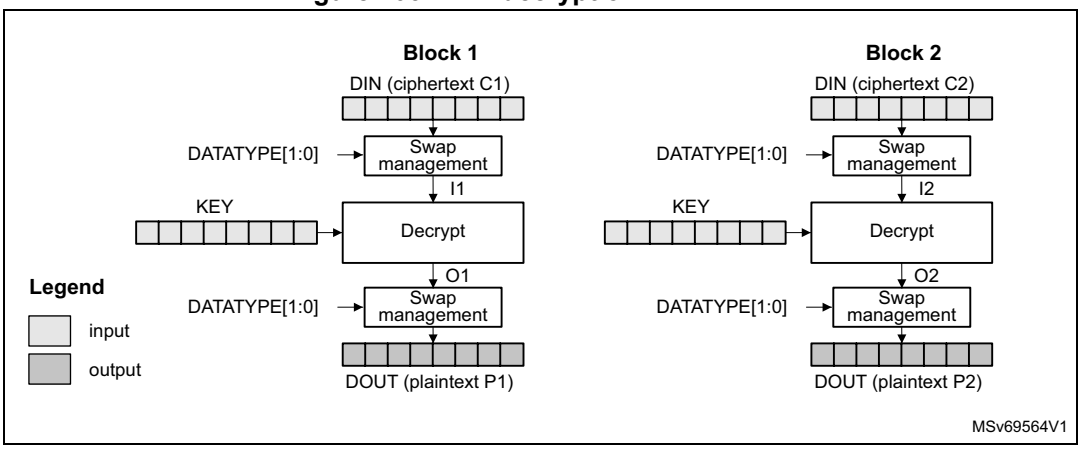


Figure 403. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 404 and *Figure 405* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[3:0] with 0x1.

Figure 404. CBC encryption

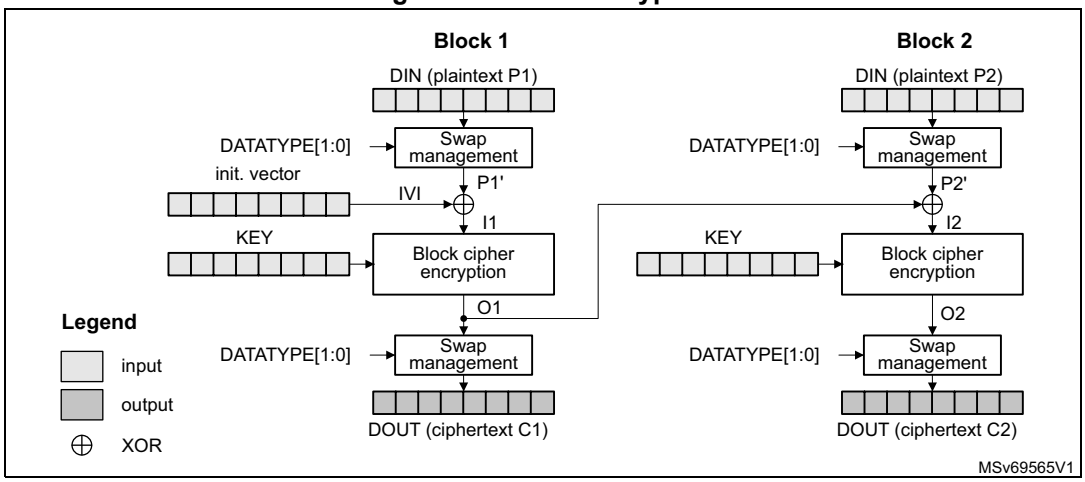
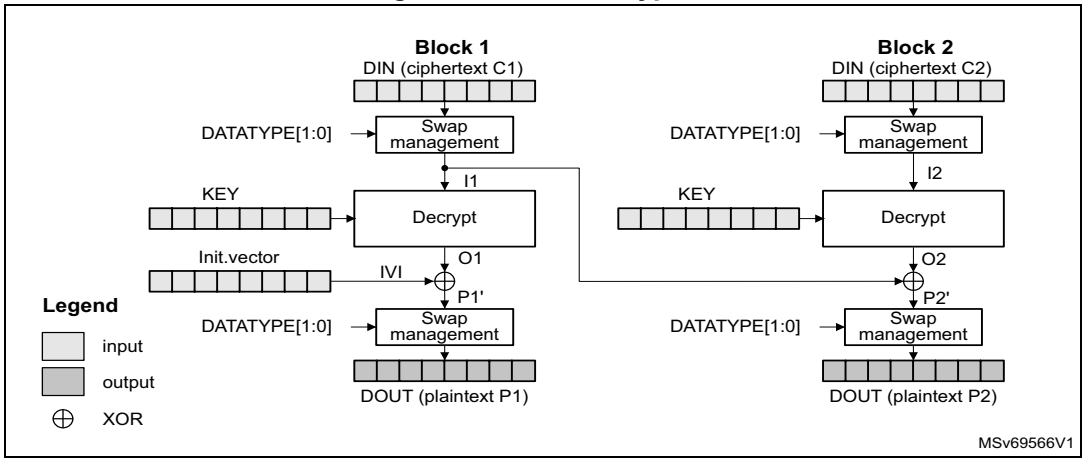


Figure 405. CBC decryption



For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   a) Select ECB or CBC chaining mode (write CHMOD[3:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   b) Configure the data type, through DATATYPE[1:0].
   c) Configure the key size, through KEYSIZE[1:0]. Also set KEYPROT if the key is not shared with a different security context (different secure or Compartment ID attribute).
   d) Select the key source using KEYSEL[2:0] (see step *5.*)
   e) Select normal key mode, by clearing WRAPEN. For wrapped key usage (with WRAPEN set) refer to *Section 48.4.14*.
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17: SAES key registers* for details).
6. Wait until KEYVALID is set (key loading complete).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 48.4.7*.
   b) Append the cleartext block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   a) Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[3:0] bitfield is not significant during this operation.
   b) Configure the data type, through DATATYPE[1:0].
   c) Configure the key size, through KEYSIZE[1:0]. Also set KEYPROT if the key is not shared with a different security context (different secure or Compartment ID attribute).
   d) Select the key source using KEYSEL[2:0] (see step *4.*)
   e) Select normal key mode, by clearing WRAPEN. For wrapped key usage (with WRAPEN set) refer to *Section 48.4.14*.
4. Write the key into the SAES_KEYRx registers. Alternatively, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17: SAES key registers* for details).
5. Wait until KEYVALID is set (key loading complete).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag of the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[3:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 48.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the cleartext block.
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral by clearing EN. Use IPRST to clear the ECB or CBC encryption key modified by the key preparation process.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (WRAPEN = 0).

Note: Suspend and resume operations are only supported when SAES is used in CPU mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation complete).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE[1:0]) then restore the SAES_KEYRx registers. Alternatively, select a key source different from key registers, using KEYSEL[2:0]. Refer to *Section 48.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

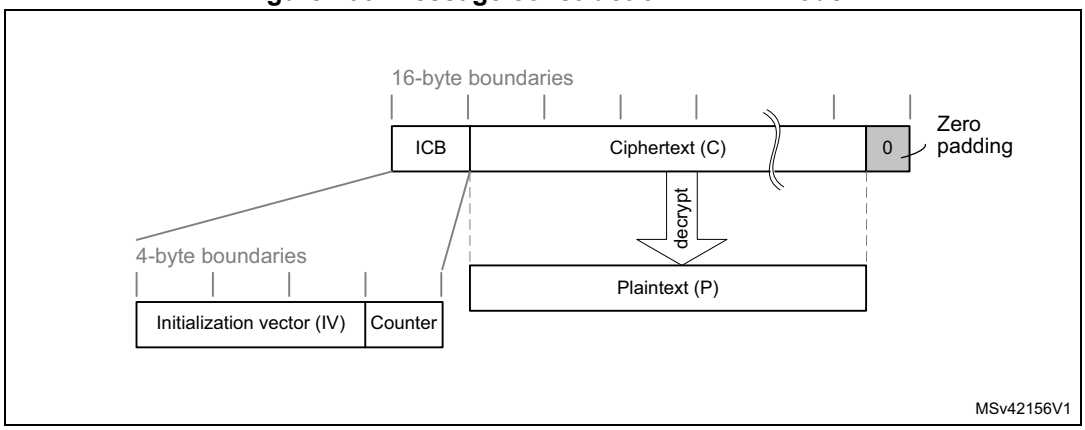
Note: It is not required to save the key registers as the application knows the original key.

48.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 406*.

Figure 406. Message construction in CTR mode



The structure of this message is:

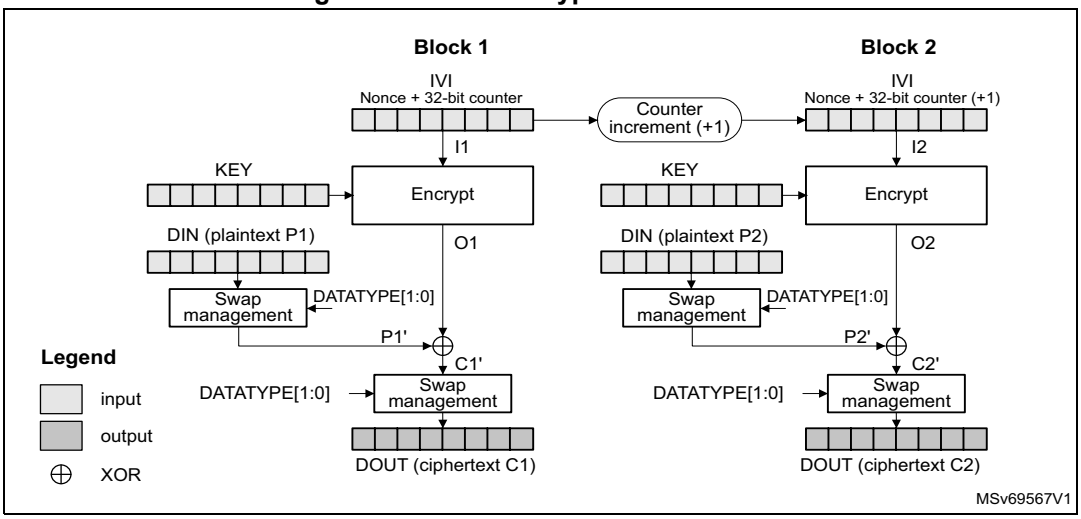
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is complete. The initial value of the counter is defined by the application.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be nonmultiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 407 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[3:0] with 0x2.

Figure 407. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 441*.

Table 441. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter

CTR encryption and decryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   a) Select CTR chaining mode (write CHMOD[3:0] with 0x2) in *encryption* or *decryption* mode (write MODE[1:0] with 0x0 or 0x2).
   b) Configure the data type, through DATATYPE[1:0].
   c) Configure the key size, through KEYSIZE[1:0]. Also set KEYPROT if the key is not shared with a different security context (different secure or Compartment ID attribute).
   d) Select the key source using KEYSEL[2:0] (see step *5.*)
   e) Select normal key mode, by clearing WRAPEN. For wrapped key usage (with WRAPEN set) refer to *Section 48.4.14*.
4. Write the initialization vector in SAES_IVRx registers according to *Table 441*.
5. Write the key into the SAES_KEYRx registers. Alternatively, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17: SAES key registers* for details).
6. Wait until KEYVALID is set (the key loading complete).
7. Enable the SAES peripheral, by setting EN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 48.4.9: SAES basic chaining modes (ECB, CBC)*. The IV registers must be saved in suspend, then restored in resume operation, because they are altered during the data processing.

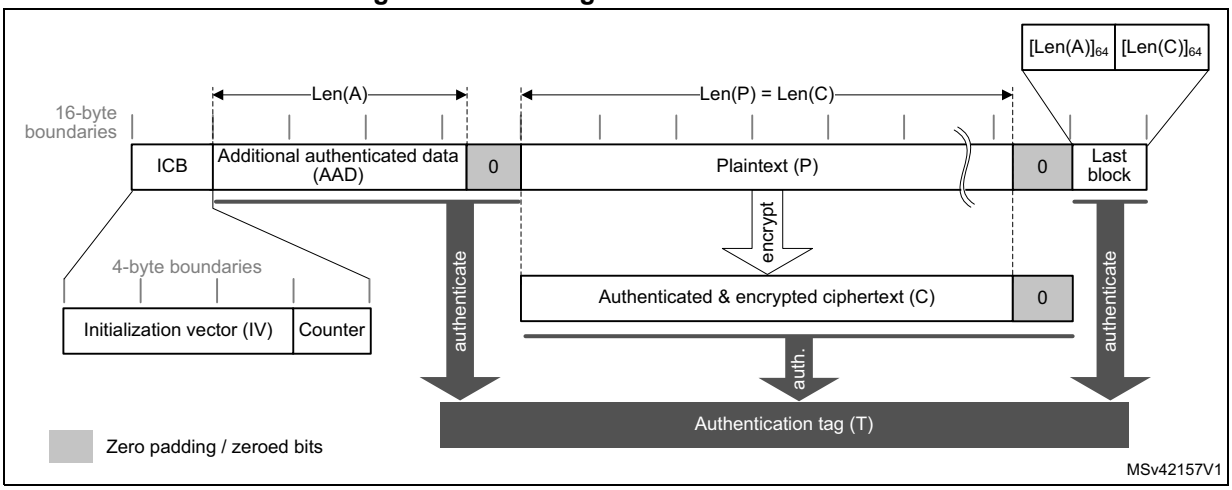
Note: Suspend and resume operations are only supported when SAES is used in CPU mode.

48.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 408. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is complete. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a nonmultiple of 16 bytes, and must not exceed 2^{64} – 1 bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be nonmultiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 443*.

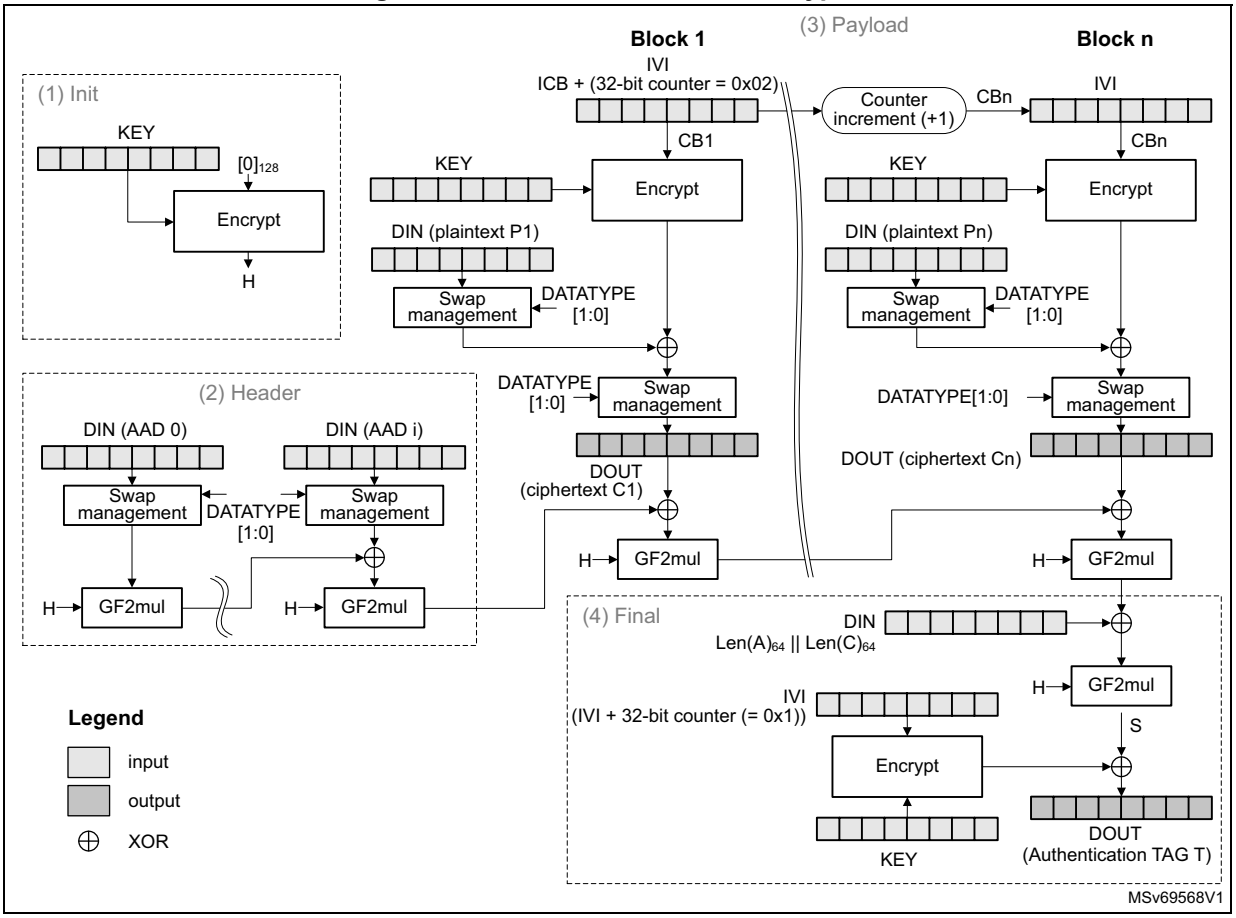
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a nonmultiple of 16 bytes a special padding scheme is required.

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 409 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[3:0] with 0x3.

Figure 409. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 442*.

Table 442. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header and the payload lengths in bits, as shown in *Table 443*.

Table 443. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 48.4.6*, with the following sequence of events.

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   a) Select GCM chaining mode (write CHMOD[3:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   b) Configure the data type, through DATATYPE[1:0]
   c) Configure the key size, through KEYSIZE[1:0]. Also set KEYPROT if the key is not shared with a different security context (different secure or Compartment ID attribute).
   d) Select the key source using KEYSEL[2:0] (see step *5.*)
   e) Select normal key mode, by clearing WRAPEN. For wrapped key usage (with WRAPEN set) refer to *Section 48.4.14*.
   f) Select the GCM initialization phase, by writing CPHASE[1:0] with 0x0
4. Write the initialization vector in SAES_IVRx registers according to *Table 442*.
5. Write the key into the SAES_KEYRx registers. Alternatively, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17: SAES key registers* for details).
6. Wait until KEYVALID is set (key loading complete).
7. Set EN to start the calculation of the hash key.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is complete.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to CPHASE[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block. As mentioned in *Section 48.4.5*, the use of NPBLB[3:0] might be necessary.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.
14. Wait until the BUSY flag clears (encryption only).

Note: This phase can be skipped if there is no payload, that is, Len(C) = 0 (see GMAC mode).

GCM finalization

15. Select the GCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during GCM initialization. If DMA was used previously, clear the DMAINEN and DMAOUTEN bits to switch to CPU mode.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 443*.
17. Wait until the CCF flag of the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit in SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

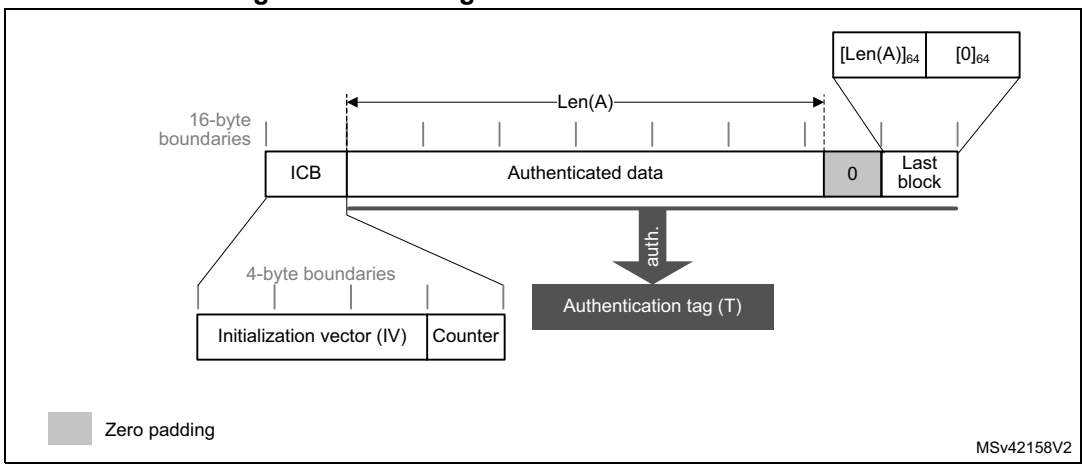
Suspend/resume operations are not supported in GCM mode.

48.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

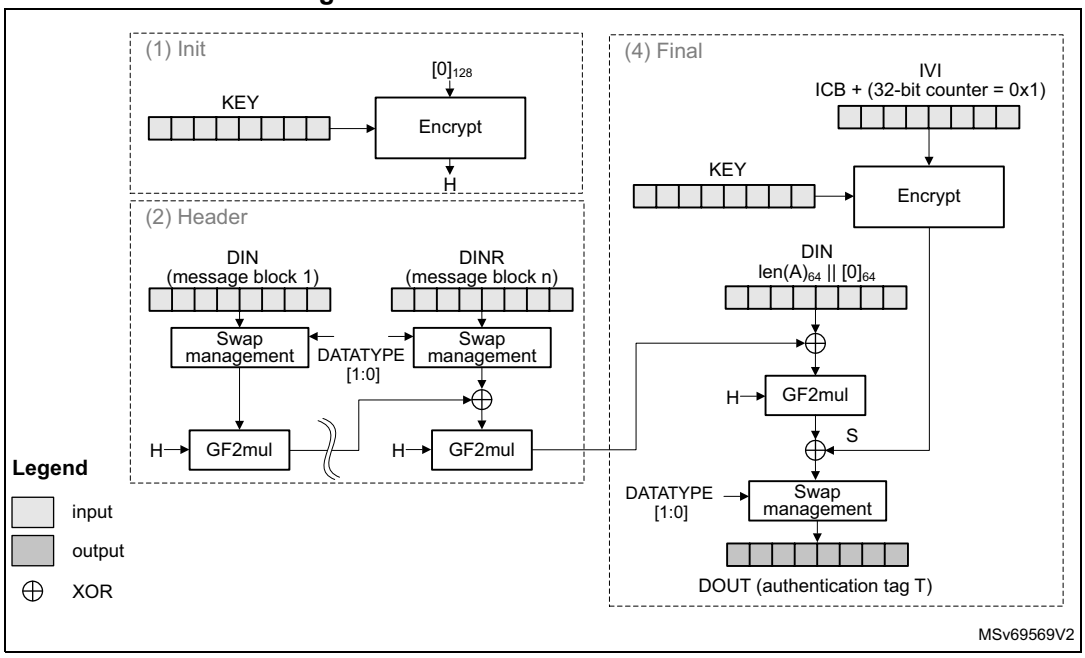
Figure 410. Message construction in GMAC mode



For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 411 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[3:0] with 0x3.

Figure 411. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC mode

Suspend/resume operations are not supported in GMAC mode.

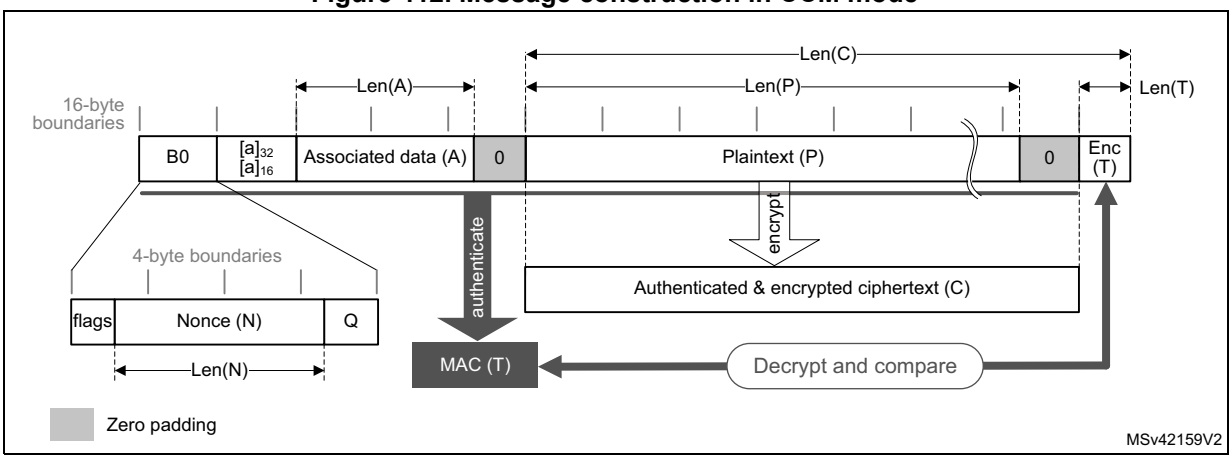
48.4.13 SAES counter with CBC-MAC (CCM)

The AES **counter with cipher block chaining-message authentication code** (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter chaining mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 412. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a nonmultiple of 16 bytes (see *Figure 412*). The

standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:

– If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
– If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
– If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on 10 bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a nonmultiple of 16 bytes (see *Figure 412*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a nonmultiple of 16 bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) is 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) is 64 bits or 8 bytes)

P: 20212223 (Len(P) is 32 bits or 4 bytes)

T: 6084341B (Len(T) is 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

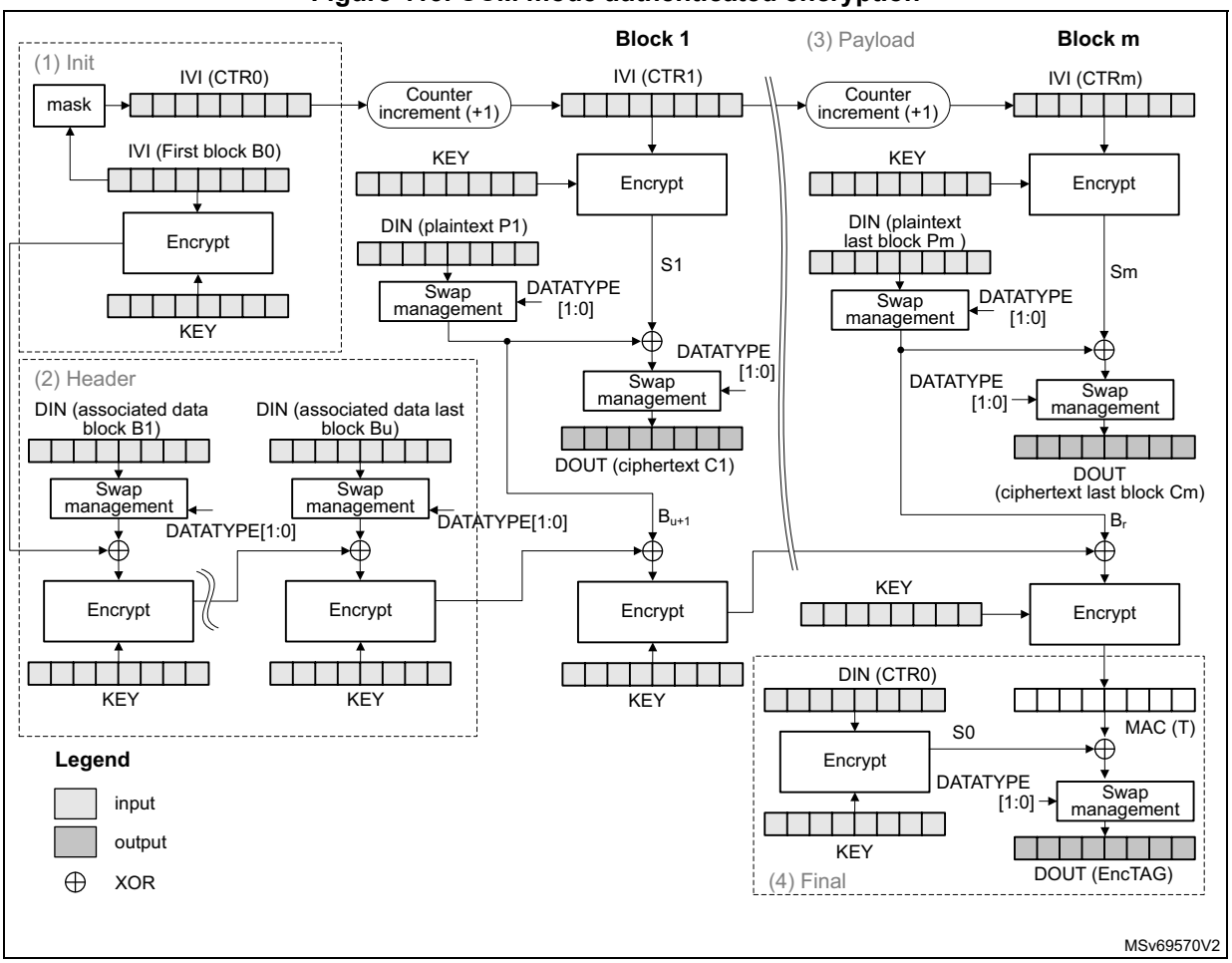
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 413 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[3:0] with 0x4.

Figure 413. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 444*.

Table 444. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 48.4.6*, with the following sequence of events.

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   a) Select CCM chaining mode (write CHMOD[3:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   b) Configure the data type, through DATATYPE[1:0].
   c) Configure the key size, through KEYSIZE[1:0]. Also set KEYPROT if the key is not shared with a different security context (different secure or Compartment ID attribute).
   d) Select the key source using KEYSEL[2:0] (see step *5.*)
   e) Select normal key mode, by clearing WRAPEN. For wrapped key usage (with WRAPEN set) refer to *Section 48.4.14*.
   f) Select the CCM initialization phase, by writing CPHASE[1:0] with 0x0.
4. Write the B0 data into the SAES_IVRx registers according to *Table 444*.
5. Write the key into the SAES_KEYRx registers Alternatively, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17: SAES key registers* for details).
6. Wait until KEYVALID is set (the key loading complete).
7. Set EN to start the first mask calculation. The EN bit clears automatically upon completion of the calculation.
8. Wait until the CCF flag of the SAES_ISR register is set.
9. Clear the CCF flag , by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing CPHASE[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block. As mentioned in *Section 48.4.5*, the use of NPBLB[3:0] might be necessary.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during the CCM initialization. If DMA was used previously, clear the DMAINEN and DMAOUTEN bits to switch to CPU mode.
15. Wait until CCF flag of the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

Note: Suspend and resume operations are only supported when SAES is used in CPU mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation complete).
2. Read the SAES_DOUTR register four times to save the last processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. In the payload phase, save the IV registers as they are altered during the data processing. This step is not required in the header phase.
6. Disable the SAES peripheral, by clearing EN.

7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. Alternatively, select a key source different from the key registers, by setting KEYSEL[2:0] to a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
4. In the payload phase, restore the SAES_IVRx registers using the saved configuration. In the header phase, write the initial setting values back to the SAES_IVRx registers.
5. Enable the SAES peripheral, by setting EN.

48.4.14 SAES basic operations with wrapped keys

SAES peripheral can wrap (encrypt) and unwrap (decrypt) the application keys using hardware-secret key (DHUK, BHK). With this feature, keys used in SAES can be made usable by the application software without being exposed in clear-text (unencrypted).

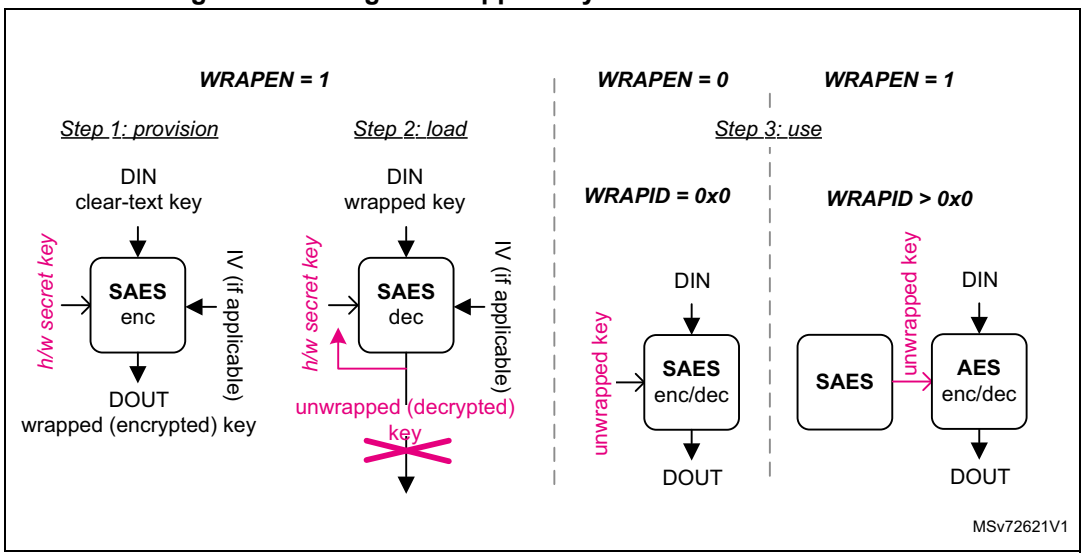
Wrapped key sequences are too small to be suspended/resumed. They also cannot be used when GCM or CCM mode are selected.

Using wrapped keys with ECB/CBC chaining modes

Figure 414 summarizes how to use wrapped keys when ECB or CBC mode are chosen.

- In Step 1, the clear-text key is encrypted once in SAES. KEYSEL[2:0] is different from zero.
- In Step 2, the encrypted key is decrypted in SAES each time the unwrapped (decrypted) key needs to be loaded in key registers.
  - Set WRAPID[2:0] to zero if the wrapped key is used in SAES
  - Set WRAPID[2:0] to a value different from zero if the wrapped key is used outside of the SAES.
- In Step 3, the target peripheral (SAES or another AES peripheral) can use the unwrapped (decrypted) key. If SAES is the target peripheral, WRAPEN must be cleared, otherwise it remains set.

Figure 414. Usage of wrapped keys in ECB and CBC modes



Provisioning wrapped keys

The recommended sequence to wrap (that is, encrypt) a key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   a) Select ECB or CBC chaining mode (CHMOD[3:0] at 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0).
   b) Select 32-bit data type (DATATYPE at 0x0).
   c) Configure the key size with KEYSIZE[1:0]. This information is used both for the encryption key and for the key to wrap/encrypt, unless KEYSIZE[1:0] is 0x1. In this case the encryption key is fixed to 256-bit, not 192-bit.
   d) Select the correct key source by using KEYSEL[2:0].
   e) Set WRAPEN and select the wrapped key use with WRAPID[2:0]. Refer to *Section 48.8.1* for details on WRAPID[2:0] options.
4. Write the initialization vector in SAES_IVRx registers if CBC mode is selected in the previous step.
5. If required, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17*).
6. Wait until KEYVALID is set (key loading complete).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first).
   If KEYSIZE[1:0] is 0x1 the 128-bit MSB block of the key to encrypt is made of two MSB words of the key (as two 32-bit LSB words), with the two 32-bit MSB words made of ciphertext stealing data (for example 8-bytes MSB of the 128 least significant key block).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps 8 to 10 if KEYSIZE[1:0] is 0x1 or 0x2.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption with WRAPEN set is only supported when ECB or CBC is selected.

Unwrapping keys in appropriate target

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   a) Select the *key derivation* mode, writing MODE[1:0] with 0x1. The CHMOD[3:0] bitfield must be the same as when the key were wrapped/encrypted.
   b) Select 32-bit data type (DATATYPE at 0x0).
   c) The KEYSIZE[1:0] bitfield must be the same as when the key were wrapped/encrypted. When KEYSIZE[1:0] is 0x1, the decryption key size is fixed to 256-bit, not 192-bit.
   d) Select with KEYSEL[2:0] the same key source as when the key was wrapped or encrypted.
   e) Set WRAPEN, with WRAPID[2:0] configured to match the value used when the key was wrapped or encrypted.
4. If required, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17*).
5. Wait until KEYVALID is set (key loading complete).
6. Enable the SAES peripheral, by setting EN.
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the decryption mode, by writing MODE[1:0] with 0x2. Do not change other parameters in SAES_CR register.
10. Write in SAES_IVRx registers the same initialization vector as when the key were wrapped/encrypted, if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps 12 and 13 if KEYSIZE[1:0]is 0x1 or 0x2.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable when the WRAPEN bit clears, based on the WRAPID value:

- If the WRAPID[2:0] value is zero, the key can be shared with an application running in a different security context (such as secure, nonsecure, specific CPU) if KEYPROT bit was cleared during the step *3.*
- If the WRAPID[2:0] value is nonzero, the application must load the key to the target AES engine as described in *Section 48.4.15*.
- At this stage, with WRAPEN and KEYVALID set, if WRAPID[2:0] value is changed from zero to nonzero, the SAES sets the KEIF error flag, thus clearing key registers and causing any key transfer sequence to fail (see *Section 48.4.19* for details).

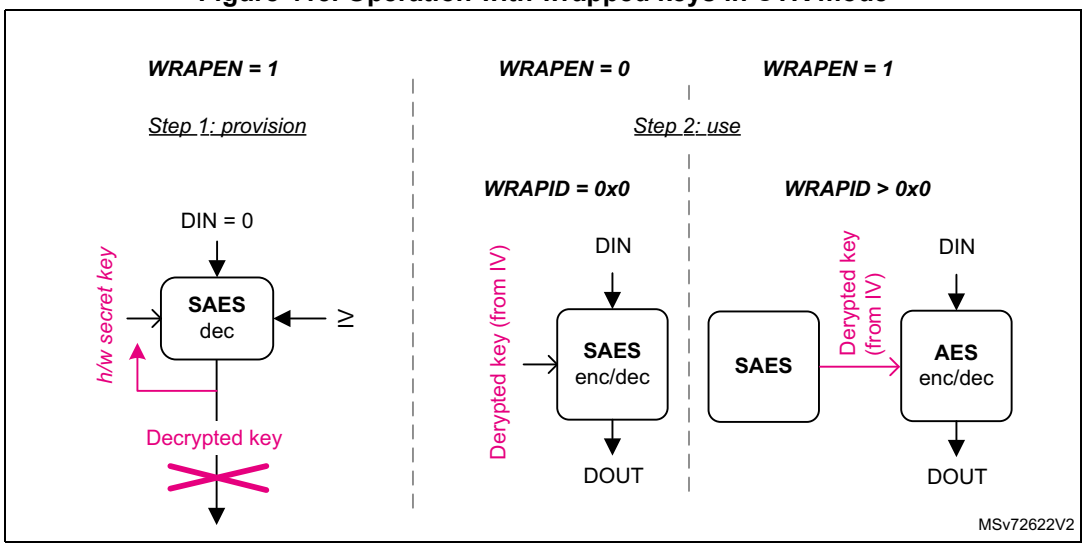
Note: When the application selects the DHUK as key source (KEYSEL[2:0] is 0x1 or 0x4), the application must use the same privilege, security, CPU, HDPL, WRAPEN/WRAPID, KEYSEL[2:0], CHMOD[3:0] and KEYSIZE[1:0] context for encryption and decryption. Otherwise, the result is incorrect.

Using wrapped keys with CTR chaining mode

Figure 415 summarizes how to use wrapped keys when CTR mode is chosen.

- In Step 1 the hardware secret key is computed in SAES each time it needs to be used (AES CTR decryption with DIN=0).
  - Set WRAPID[2:0] to zero if wrapped key is used in SAES
  - Set WRAPID[2:0] to a value different from zero if wrapped key is used outside of the SAES.
- In Step 2 the target peripheral (SAES or another AES peripheral) can use the computed key. If SAES is the target peripheral, WRAPEN must be cleared, otherwise it remains set.

Figure 415. Operation with wrapped keys in CTR mode



Recommended sequence for SAES wrapped key usage with CTR is described below:

1. Disable the SAES peripheral, clearing EN.
2. Wait until BUSY clears (indicating no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   a) Select the CTR chaining mode (CHMOD[3:0] at 0x2) in *decryption* mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   b) Select 32-bit data type (DATATYPE at 0x0).
   c) Configure the required key size using KEYSIZE[1:0]. 192-bit key (KEYSIZE[1:0] is 0x1) is not supported.
   d) Select the correct key source by using KEYSEL[2:0].
   e) Set WRAPEN and select the wrapped key usage with WRAPID[2:0]. Refer to *Section 48.8.1* for details on WRAPID[2:0] options.
4. Write the initialization vector in SAES_IVRx registers. This information can be the ID of the key.
5. If required, manage the key source selected by KEYSEL[2:0] (see *Section 48.4.17*).
6. Wait until KEYVALID is set (key loading complete).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_SR register. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
9. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable when the WRAPEN bit clears, based on the WRAPID value:

- If the WRAPID[2:0] value is zero, the key can be shared with an application running in a different security context (such as secure, nonsecure, specific CPU) if the KEYPROT bit was cleared during the step *3.*
- If WRAPID[2:0] is different from zero, the application must load the key to the target AES engine as described in next *Section 48.4.15*.
- At this stage, with WRAPEN and KEYVALID set, if WRAPID[2:0] value is changed from zero to nonzero, SAES sets KEIF error flag, thus clearing key registers and causing any key transfer sequence to fail (see *Section 48.4.19* for details).

Note: When the application selects the DHUK as key source (KEYSEL[2:0] is 0x1 or 0x4), the application must use the same privilege, security, CPU, HDPL, WRAPEN/WRAPID, KEYSEL[2:0], CHMOD[3:0] and KEYSIZE[1:0] context for encryption and decryption. Otherwise, the result is incorrect.

Setting encryption with WRAPEN set and CTR mode selected automatically clears EN bit, disabling the peripheral.

48.4.15 Wrapped key usage outside the SAES peripheral

When WRAPID[2:0] is different from zero an AES peripheral can use the unwrapped key loaded in SAES peripheral. Refer to the WRAPID[2:0] bitfield description in *Section 48.8.1* for the supported AES peripherals. The key unwrapping procedure in SAES is described in previous *Section 48.4.14*.

Once the wrapped key is decrypted in SAES key registers, it can be shared with the target AES peripheral (exclusively), while WRAPEN and KEYVALID are set in SAES. Key sharing sequence can be run multiple times in the target AES peripheral (for example, to manage a suspend/resume situation), as long as SAES is unused and duly remains in key sharing state, with WRAPEN and KEYVALID set.

If one of the CRYP1 and CRYP2 peripherals not selected by WRAPID[2:0] tries to read the unwrapped key in SAES, the key transfer sequence fails without errors.

If KEYSIZE[1:0] value in target CRYP1 or CRYP2 peripheral differs from SAES, the key transfer sequence fails with key error flag raised in both peripherals.

Note: KEYSIZE[1:0] is 0x1 (192-bit key) is not supported when WRAPEN is set and WRAPID[2:0] is greater than 0x0.

While WRAPEN and KEYVALID are set in SAES, modifying WRAPID[2:0] triggers a KEIF error, which causes the key registers to be cleared and any key transfer sequence to fail (see *Section 48.4.19* for details).

48.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 446* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output, except in GCM final phase (in which data must be inserted normally).

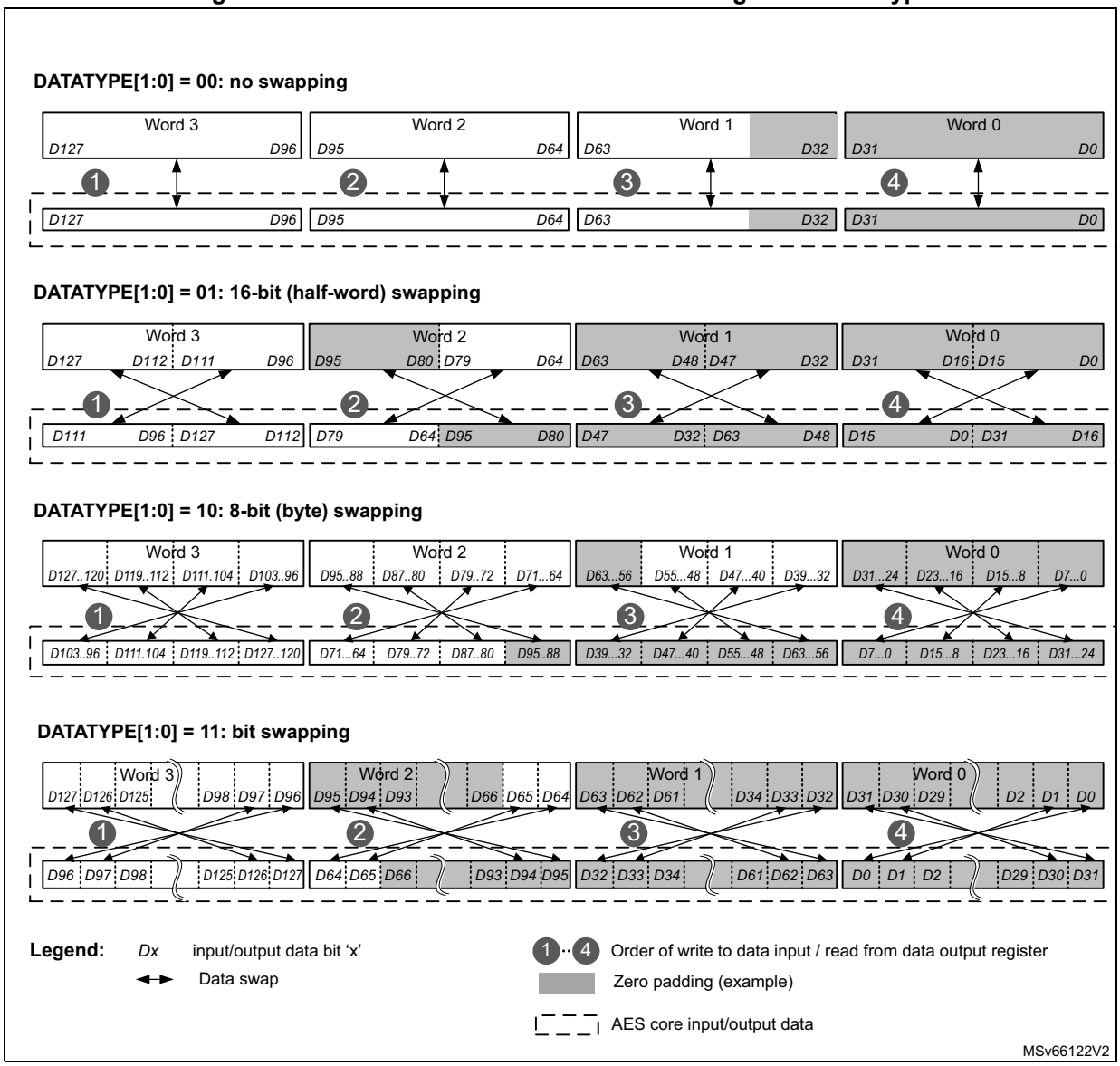
Note: The data in the SAES key registers (SAES_KEYRx) and the initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

Table 446 and *Figure 416* summarize the SAES data swapping feature. As shown, the padding bytes are swapped with the data written in SAES_DINR because the padding process is completed before the data swapping.

Table 445. SAES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 416. 128-bit block construction according to the data type



Data padding

Figure 416 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] equal to 0x0
- 48 message bits, with DATATYPE[1:0] equal to 0x1
- 56 message bits, with DATATYPE[1:0] equal to 0x2
- 34 message bits, with DATATYPE[1:0] equal to 0x3

48.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 446*. Reads are not allowed for security reason.

Note: In the memory and in the SAES key registers, keys are stored in little-endian format, with most significant byte on the highest address.

Table 446. Key endianness in SAES_KEYRx registers (128/192/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
-	-	-	-	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]
-	-	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
-	-	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when the SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. With KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x = 3 to 0) for KEYSIZE[1:0] equal to 0x0
- SAES_KEYRx (x = 0 to 5 or x = 5 to 0) for KEYSIZE[1:0] equal to 0x1
- SAES_KEYRx (x = 0 to 7 or x = 7 to 0) for KEYSIZE[1:0] equal to 0x2

Note: KEYSIZE[1:0] must be written before the key. When KEYSIZE[1:0] is 0x1 and WRAPEN is set, the application must write the eight key registers (not only six).

As soon as the first key register is written, the KEYVALID flag clears. Once the key registers writing sequence is complete, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 48.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key to the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a nonvolatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE[1:0] information. Thanks to the key derivation function, secure SAES uses a secure DHUK, while nonsecure SAES uses a nonsecure DHUK.

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into the key registers upon a secure application performing a single read in ascending order of all the TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE[1:0] equal to 0x0, x = 0 to 7 for KEYSIZE[1:0] equal to 0x1 or 0x2). Refer to *Table 446*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read in ascending order of all the TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE[1:0] equal to 0x0, x = 0 to 7 for KEYSIZE[1:0] equal to 0x1 or 0x2). Refer to *Table 446*.

KEYVALID must be cleared when the application needs to reload a key into the key registers selected by KEYSEL[2:0], provided the value is different from zero. For example, this is necessary when switching from ECB decryption to ECB encryption using the same key. When KEYVALID is set, the recommended method to clear it is by using IPRST (see *Section 48.8.1*).

For nonzero KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE[1:0] is transferred to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 48.4.19*).

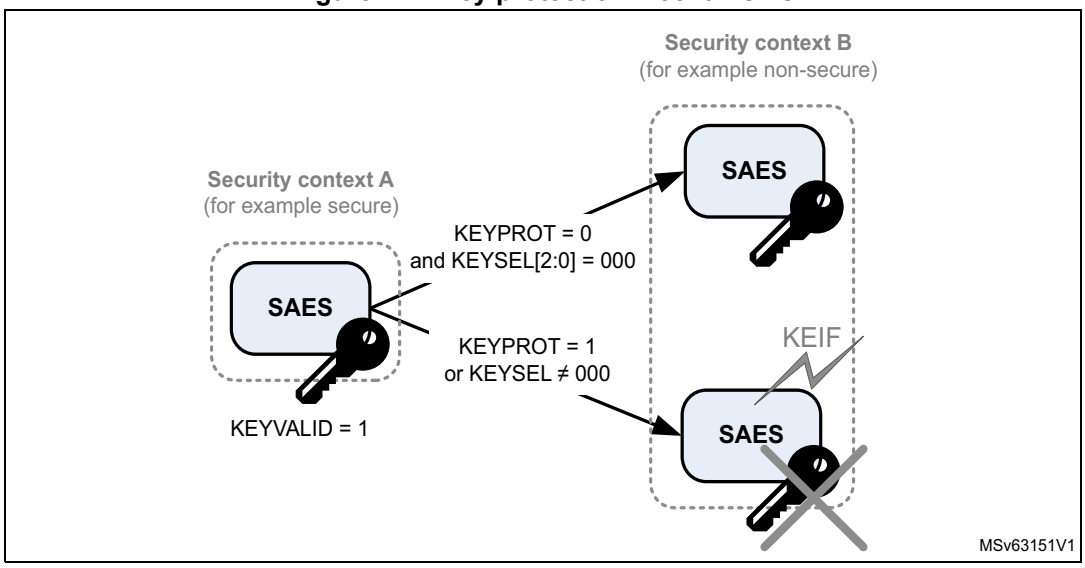
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 417*, when the application sets the KEYPROT bit before writing a key into SAES_KEYRx, any application executing in a different security context (that is, different security or Compartment ID attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 417. Key protection mechanisms



48.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 447*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In the memory and in the SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 447. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

48.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 48.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 48.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security or Compartment ID attribute) accesses the SAES peripheral.
- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 48.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when WRAPEN is set, WRAPID[2:0] is greater than 0x0, and KEYSIZE[1:0] in CRYP1 or CRYP2 peripheral does not match KEYSIZE[1:0] in the SAES peripheral.
- **Key sharing error:** triggered upon a failure to transfer the SAES shared key to the CRYP1 or CRYP2 peripheral. See *Section 48.4.15: Wrapped key usage outside the SAES peripheral* for details.
- **Hardware secret key loading error:** triggered upon failing to load DHUK or BHK into SAES. The corresponding KEYSEL[2:0] value is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 48.5: SAES interrupts*. Activating interrupts on key errors is recommended, for security reason.

Upon a key selection error, use IPRST to clear KEIF and the source of error (specific value of KEYSEL[2:0] and KEYSIZE[1:0]). Persisting problems may require a power-on reset.

Upon a key sharing error, reset both CRYP1 (or CRYP2, depending on the key sharing target) and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and reconfiguring SAES. Clearing KEIF without updating KEYSEL[2:0] and KEYSIZE[1:0] restarts the key loading process.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 48.5: SAES interrupts*.

Upon an RNG error:

- Verify that the RNG peripheral is properly initialized, its AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

About DPA errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any processing in SAES.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

48.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- Computation complete (CCF)
- Read error (RDERRF)
- Wwrite error (WRERRF)
- Key error (KEIF)
- RNG error (RNGEIF)

See *Section 48.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. The application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 448 gives a summary of the available features.

Table 448. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation complete flag	CCF	CCFIE	set CCF(1)
	read error flag	RWEIF, RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	RWEIF, WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flags in the SAES_SR register. Setting either of them also sets the RWEIF flag in the SAES_ISR register.

48.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests. Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in section *Appending data via DMA*.

48.6.1 Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 48.4.7).

48.6.2 Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase.

Note: According to the message size, extra bytes might need to be discarded by the application in the last block.

48.6.3 Stopping DMA transfers

All DMA request signals are deasserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

48.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode, given in SAES AHB clock cycles.

Table 449. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**82**
	Key preparation	-	**66**
192-bit	Encryption or decryption(2)	ECB, CBC, CTR	**90**
	Key preparation	-	**82**

Table 449. Processing latency for ECB, CBC and CTR (continued)

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**102**
	Key preparation	-	**88**

1. SAES kernel clock.
2. Excluding key preparation time (ECB and CBC only).

Table 450. Processing latency for GCM and CCM

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)(2)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	**103**	**73**	**82**(3)	**82**
		CCM	**93**	**86**	**175**	**88**
192-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	**111**	**73**	**90**(3)	**90**
		CCM	**101**	**94**	**191**	**96**
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	**123**	**73**	**102**(3)	**102**
		CCM	**113**	**106**	**215**	**108**

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. Value given for saes_hclk = saes_ker_ck.
3. For the last block processing time is increased to 286 cycles.

48.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

48.8.1 SAES control register (SAES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			WRAPID[2:0]			WRAPEN	NPBLB[3:0]				KEYPROT	KEYSIZE[1:0]		Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	CHMOD[3:0]				MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES software reset

Setting this bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself, and the RNGEIF bit. Hence, any key-relative data is lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be low while writing any configuration registers. After setting this bit, the application must clear it when BUSY clears in SAES_SR (IPRST clear is ignored otherwise).

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Other: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY is cleared and KEYVALID set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID clears.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, KEYSEL[2:0] is cleared if WRAPEN is set.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, nonsecure, specific CPU). More specifically, when security or Compartment ID of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:25 **WRAPID[2:0]**: Wrap ID

This bitfield defines, at the end of a decryption process with WRAPEN set, which target can use the key stored in SAES key registers.

0x0: The SAES itself

0x1: The CRYP1 peripheral, through the key bus

0x2: The CRYP2 peripheral, through the key bus

Other: Reserved (do not use)

When WRAPEN is set and WRAPID is different from zero, the SAES key registers are shared with the peripheral selected with this bitfield after a successful decryption process. This sharing ends when either WRAPEN or KEYVALID is cleared. Changing this bitfield value when WRAPEN and KEYVALID are set triggers a key error (KEIF set in SAES_ISR), with KEYVALID automatically cleared.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 24 **WRAPEN**: Wrap enable

Set this bit when manipulating wrapped (or encrypted) keys.

0: Normal key. The key registers are freely usable and no specific use or protection applies to the SAES_DINR and SAES_DOUTR registers.

1: Wrapped key. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into the SAES key registers, after a successful decryption process.

When WRAPEN is set WRAPID[2:0] and KEYSIZE[1:0] must be properly initialized.

When WRAPEN is set any attempt to configure the SAES peripheral for use by an application belonging to a different security domain (such as secure or nonsecure, other CPU) results in automatic key erasure and setting of the KEIF flag.

Attempts to write this bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0] is 0x0, the application can transfer the ownership of SAES, with its loaded key, to an application running in another security context (such as nonsecure, secure, specific CPU).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, nonsecure, Compartment ID) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 18:17 **KEYSIZE[1:0]**: Key size

This bitfield defines the key length in bits of the key used by SAES.

0x0: 128-bit

0x1: 192-bit

0x2: 256-bit

0x3: Reserved

When WRAPEN is set, KEYSIZE[1:0] also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **CPHASE[1:0]:** Chaining phase

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes. It is ignored when ECB, CBC or CTR is selected, and WRAPEN is set.

0x0: Initialization phase

0x1: Header phase

0x2: Payload phase

0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.

0: Disable

1: Enable

Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.

0: Disable

1: Enable

Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:9 Reserved, must be kept at reset value.

Bits 8:5 **CHMOD[3:0]**: Chaining mode

This bitfield selects the AES chaining mode in SAES.

0x0: Electronic codebook (ECB)

0x1: Cipher-Block Chaining (CBC)

0x2: Counter Mode (CTR)

0x3: Galois Counter Mode (GCM) and Galois Message Authentication Code (GMAC)

0x4: Counter with CBC-MAC (CCM)

Other: Reserved

Attempts to write the bitfield are ignored when the BUSY bit of the SAES_SR register is set, as well as when the EN bit of this register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:

0x0: Encryption

0x1: Key derivation (or key preparation), for ECB/CBC decryption only

0x2: Decryption

0x3: Reserved

Attempts to write the bitfield are ignored when the BUSY bit of the SAES_SR register is set, as well as when the EN bit of this register is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written to the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 48.4.16: SAES data registers and data swapping*.

0x0: No swapping (32-bit data)

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When the WRAPEN and the BUSY bits are set, IPRST bit is used instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared. It cannot be set either when SAES is in one of the following configurations:

– WRAPEN set, CHMOD[3:0] at 0x3 (GCM)

– WRAPEN set, CHMOD[3:0] at 0x2 (CTR), and MODE[1:0] at 0x0 (encryption).

48.8.2 SAES status register (SAES_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	WRERRF	RDERRF	BUSY	Res.	Res.	Res.	Res.
								r	r	r	r				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE[1:0] is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written to the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 48.4.17: SAES key registers*.

Bit 6 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 5 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 4 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (EN = 0) or when the last processing is complete.

SAES is flagged as busy when processing a block data, fetching random number from the RNG, transferring a shared key to the target peripheral, or preparing a key (ECB or CBC decryption only).

The GCM final phase must only be selected when BUSY is low.

Bits 3:0 Reserved, must be kept at reset value.

48.8.3 SAES data input register (SAES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

48.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Read when WRAPEN is set while MODE[1:0] is at 0x2 and EN is set triggers a read error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.
Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].
Writes are ignored.

48.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.
Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0, writes to the key registers are ignored, and the KEIF bit in the SAES_ISR register is set without changing KEYVALID.
With WRAPEN cleared, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

48.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.9 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.10 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.11 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.12 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.13 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register.

48.8.14 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.15 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.16 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.17 SAES context swap registers (SAES_SUSPRx)

Address offset: 0x050 + x * 0x4, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSPx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSPx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSPx[31:0]**: Context swap data x

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 48.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit in SAES_CR sets this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM, or when SAES is used in DMA mode.

48.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE	Res.	Res.
										rw	rw	rw	rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RNGEIE:** RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 4 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 3 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (Read or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 2 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bits 1:0 Reserved, must be kept at reset value.

48.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF	Res.	Res.
										r	r	r	r		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 4 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in SAES_SR.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] is 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] is 0x2 or 0x4).

–CRYP1 or CRYP2 fails to load the key shared by the SAES peripheral (WRAPEN set and WRAPID is 0x1 or 0x2, respectively).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE[1:0] at 0x0, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or the reverse. For KEYSIZE[1:0] at 0x2, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or the reverse. For KEYSIZE[1:0] at 0x1, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5, or the reverse).

–When WRAPEN is set, WRAPID[2:0] value is changed after KEYVALID is set.

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 3 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 2 **CCF:** Computation complete flag

This flag indicates whether the computation is complete. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not complete

1: Complete

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

Bits 1:0 Reserved, must be kept at reset value.

48.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF	Res.	Res.
										w	w	w	w		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RNGEIF:** RNG error interrupt flag clear

Setting this bit clears the RNGEIF status bit in SAES_ISR register.

Bit 4 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit in SAES_ISR register.

Bit 3 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 2 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit in SAES_ISR register.

Bits 1:0 Reserved, must be kept at reset value.

48.8.21 SAES hardware configuration register (SAES_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG1[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CFG1[3:0]**: HW Generic 1
This field returns 0x1.

48.8.22 SAES version register (SAES_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
This bitfield returns the SAES peripheral major version (0x1).

Bits 3:0 **MINREV[3:0]**: Minor revision
This bitfield returns the SAES peripheral minor version (0x0).

48.8.23 SAES identification register (SAES_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code of the peripheral
This bitfield returns the identification code of the SAES peripheral.

48.8.24 SAES size ID register (SAES_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code
This bitfield returns the size identification code of the SAES peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1 Kbyte address decoding)

48.8.25 SAES register map

Table 451. SAES register map and reset values

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			WRAPID[2:0]			WRAPEN	NPBLB[3:0]				KEYPROT	KEYSIZE[1:0]		Res.	Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	CHMOD[3:0]				MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0			0	0	0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	WRERRF	RDERRF	BUSY	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x008	SAES_DINR	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DATAOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 451. SAES register map and reset values (continued)

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR0	SUSP0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR1	SUSP1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x068	SAES_SUSPR6	SUSP6[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	SAES_SUSPR7	SUSP7[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE	Res.	Res.
	Reset value																											0	0	0	0		
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF	Res.	Res.
	Reset value																											0	0	0	0		
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF	Res.	Res.
	Reset value																											0	0	0	0		

Table 451. SAES register map and reset values (continued)

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F0	SAES_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG1[3:0]			
	Reset value																													0	0	0	1
0x3F4	SAES_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	SAES_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0x3FC	SAES_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4 on page 184* for the register boundary addresses.

49 Cryptographic processor (CRYP)

49.1 CRYP introduction

The cryptographic processor (CRYP) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It is also fully compliant with the data encryption standard (DES) and Triple-DES (TDES) defined by NIST and ANSI.

CRYP supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128, 192 or 256 bits in AES mode. In DES/TDES mode it supports ECB and CBC chaining modes for key size of 64-bit (including parity). CRYP has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports both single and fixed DMA burst transfers for incoming and outgoing data (two DMA channels are required). CRYP also includes input and output FIFOs for better performance.

49.2 CRYP main features

AES features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode, with a special AES-STS variant
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 16-byte data block processing, supporting cipher key lengths of 128, 192 and 256 bits
  - 14 or 18 clock cycle latency in ECB mode for processing one 16-byte block with 128-bit or 256-bit key, respectively
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption

DES/TDES features

- Compliant with NIST FIPS publication 46-3 *"Data Encryption Standard (DES)" and* ANSI *X9.52 "Triple Data Encryption Algorithm Modes of Operation (TDES)"*
- Encryption and decryption with support of ECB and CBC chaining modes
- 8-byte data block processing, supporting cipher key lengths of 64, 128 and 192 bits (including parity)
  - 16 (respectively 48) clock cycles for processing one 8-byte block of data in DES (respectively TDES) ECB mode
- Direct implementation of simple DES algorithms (a single key K1 is used)

Features common to DES/TDES and AES

- 256-bit of write-only registers for storing the cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit input buffer associated with an internal input FIFO of eight 32-bit words, corresponding to four incoming DES blocks or two AES blocks
- 32-bit output buffer associated with an internal output FIFO of eight 32-bit words, corresponding to four processed DES blocks or two AES blocks
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. The output FIFO supports both single and burst transfers, while the input FIFO supports only burst transfers.
- Data swapping logic to support 1-, 8-, 16- or 32-bit data
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only. Other access types generates an AHB error, and write accesses are ignored.
- Possibility for software to suspend a message if CRYP needs to process another message with a higher priority, then resume the original message

49.3 CRYP implementation

The devices have one CRYP peripheral, implemented as per the following table.It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 452. CRYP versus SAES features

Modes or features(1)	CRYP	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	14	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and CRYP (AES mode only)	X	
Key size in bits (AES mode)	128, 192, 256	128, 256

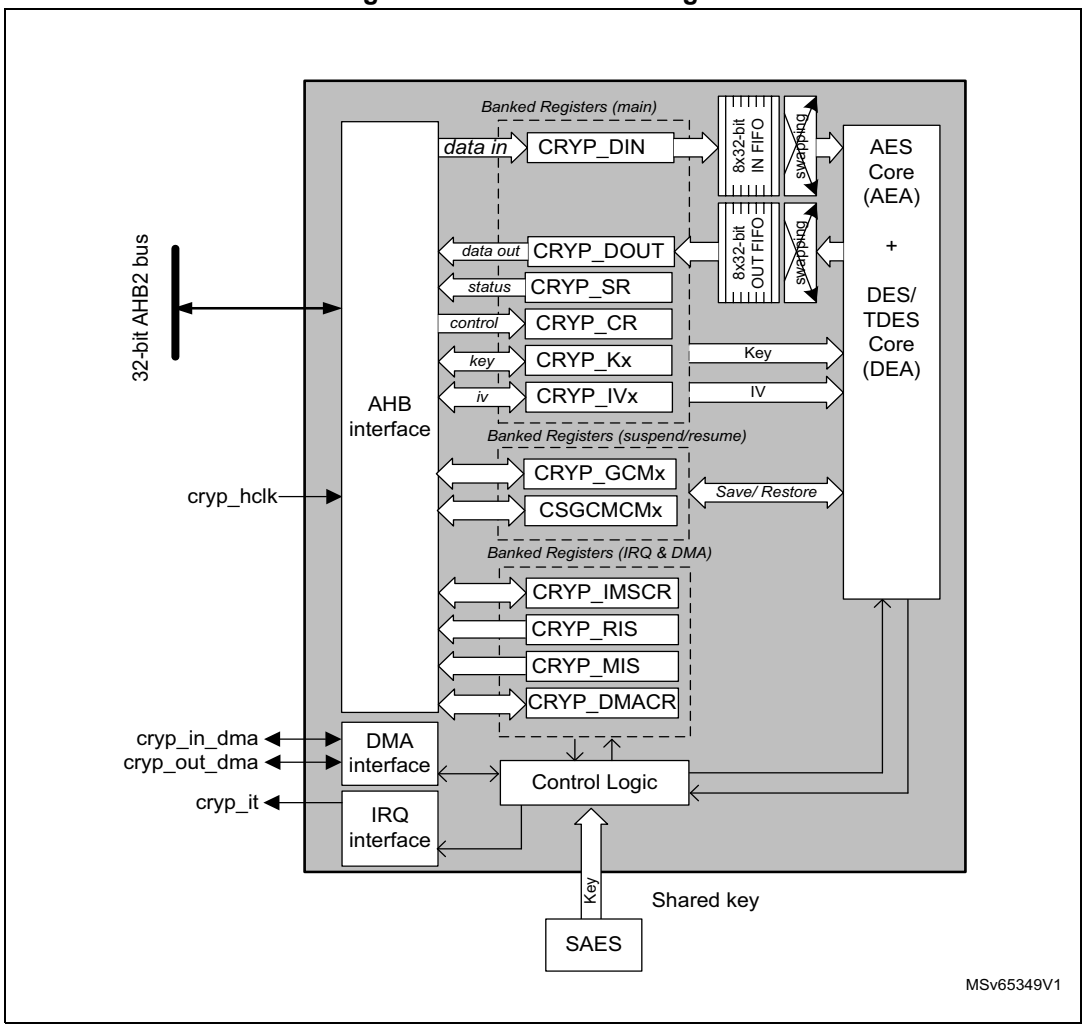
1. X = supported.

49.4 CRYP functional description

49.4.1 CRYP block diagram

The figure below shows the block diagram of the cryptographic processor.

Figure 418. CRYP block diagram



49.4.2 CRYP internal signals

Table 453 describes the user relevant internal signals interfacing the CRYP peripheral.

Table 453. CRYP internal input/output signals

Signal name	Signal type	Description
cryp_hclk	Digital input	AHB clock
cryp_it	Digital output	CRYP interrupt request
cryp_in_dma	Digital input/output	CRYP incoming data FIFO DMA request/ acknowledge
cryp_out_dma	Digital input/output	Output FIFO DMA burst request/ acknowledge (with single request for DES/TDES)

49.4.3 CRYP reset and clocks

The CRYP peripheral is clocked by the AHB clock.

The CRYP has a dedicated reset bit in the RCC.

49.4.4 CRYP symmetric cipher implementation

AES mode

The cryptographic processor (CRYP) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 454*. Those functions can be certified NIST PUB 140-3.

Table 454. CRYP approved AES symmetric key functions

Operations	Algorithm	Specification	Key size in bits	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 192, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

DES/TDES mode

The cryptographic processor (CRYP) also encrypts or decrypts 8-byte blocks of data using the data encryption standard (DES) and the triple-DES (TDES), as summarized in *Table 455*.

Table 455. CRYP approved DES/TDES symmetric key functions

Operations	Algorithm	Specification	Key size in bits	Chaining modes
Encryption, decryption	DES	FIPS PUB 46-3 ANSI X9.52	64	ECB, CBC
	TDES		64, 128, 192	

CRYP supports three keying options:

- *Keying Option 1, or 3-key TDES: K1, K2 and K3 keys are independent.*
- *Keying Option 2, or 2-key TDES: K1 and K2 are independent and K3 is equal to K1.*
- *Keying Option 3, or 1-key TDES (equivalent to single DES): K1, K2 and K3 are equal.*

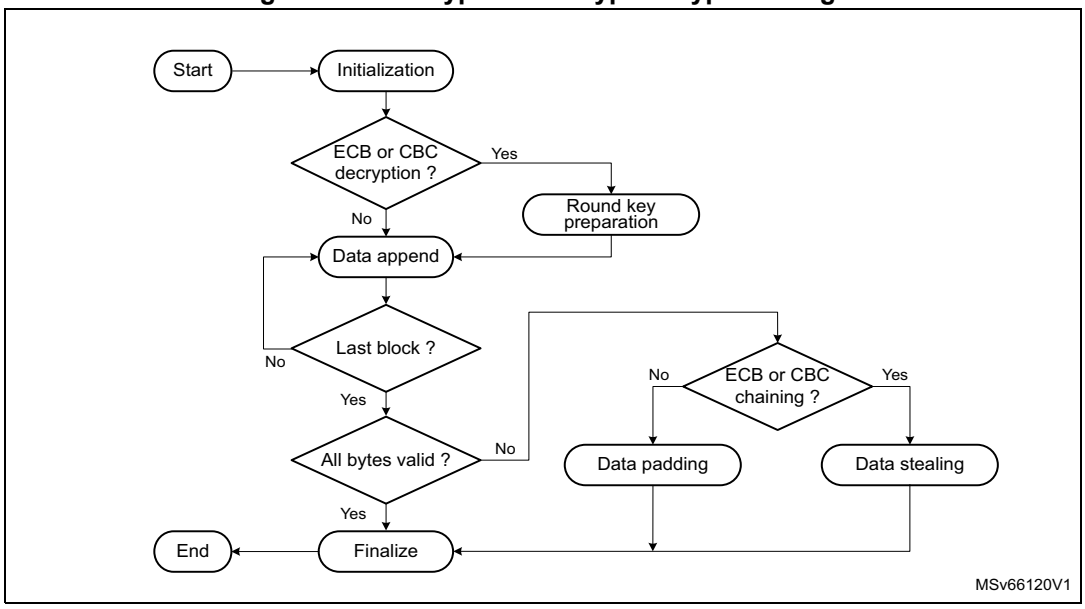
Modes common to AES and DES/TDES

CRYP can be used directly by the CPU, or indirectly using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any CRYP processing, following the sequence described in *Section 49.4.8*.

49.4.5 CRYP encryption/ decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 419. Encryption/ decryption typical usage



Initialization

The CRYP peripheral is initialized according to the chaining mode. Refer to *Section 49.4.10: CRYP AES basic chaining modes (ECB, CBC)* and *Section 49.4.11: CRYP counter mode (AES-CTR)* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 49.4.7: CRYP ciphertext stealing and data padding* if the size of data to process is not a multiple of 8 bytes (for DES/TDES) or 16 bytes (for AES). The last block management in these cases is more complex than what is described in this section.

General conditions for data processing

The CRYP is busy and processing data (BUSY flag set) when all the conditions below are met:

- CRYPEN is set in CRYP_CR register.
- There are enough data in the input FIFO (at least two words for the DES/TDES algorithm mode, four words for the AES algorithm mode).
- There is enough free-space in the output FIFO (at least two word locations for DES/TDES, four for AES).

It is possible to clear the CRYPEN bit while BUSY bit is set. In this case the ongoing DES/TDES or AES processing first completes (that is, the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

When an 8-byte block is being processed in DES/TDES mode, if the output FIFO becomes full and the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

Note: If the application needs to suspend a message to process another one with a higher priority, refer to Section 49.4.8: CRYP suspend and resume operations.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. When KEYVALID is set, enable CRYP by setting the CRYPEN bit of the CRYP_CR register (if not already done).
2. Repeat the following subsequence until the payload is entirely processed:
   a) Write data in the input FIFO (one block or until the FIFO is full).
   b) Repeat until the second last block of data has been processed:
   – Wait until the not-empty flag OFNE is set, then read the output FIFO (one block or until the FIFO is empty).
   – Wait until the not-full flag IFNF is set, then write the input FIFO (one block or until the FIFO is full), except if it is the last block.
   c) The BUSY bit is set automatically by CRYP. At the end of the processing, the BUSY bit is cleared and both FIFOs are empty (input FIFO empty flag IFEM is set, output FIFO not-empty flag OFNE is cleared).
   d) If the next processing block is the last block, pad the data with zeros to obtain a complete block (when applicable), and specify the number of non-valid bytes (using NPBLB[3:0]) in case of AES GCM payload encryption or AES CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared.
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the CRYP peripheral by clearing CRYPEN.

Note: NPBLB[3:0] bitfield is not used in header phase of AES GCM, GMAC and CCM chaining modes.

Appending data using the CPU in Interrupt mode

The method uses interrupt from the CRYP peripheral to control the data append, through the following sequence:

1. Enable interrupts from CRYP, by setting the INIM and OUTIM bits of the CRYP_IMSCR register.
2. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).
3. Handle the data in the CRYP interrupt service routine that manages the input FIFO. Upon each interrupt:
   a) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0] bitfield) in case of AES GCM payload encryption or AES CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared. After that load the block into the input FIFO.
   b) If the next processing block is not the last block, load the data into the input FIFO. Application can load only one block (2 words for DES/TDES, 4 words for AES), or load data until the input FIFO is full.
   c) After the last word of data has been written, disable the input FIFO interrupts by clearing the INIM bit (if applicable), then quit the interrupt service routine.
4. Handle the data in the CRYP interrupt service routine that manages the output FIFO. Upon each interrupt:
   a) Read data from the output FIFO. Application can read only one block (2 words for DES/TDES, 4 words for AES), or read data until the FIFO is empty.
   b) When the last word of the message has been read, INIM and BUSY bits are cleared and both FIFOs are empty (IFEM is set, OFNE is cleared). Disable the output FIFO interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.
   c) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then quit the interrupt service routine.

Note: The NPBLB[3:0] bitfield is not used in the header phase of AES GCM, GMAC and CCM chaining modes.

Appending data using the DMA

With this method, all the transfers and processing are managed by DMA and CRYP. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros in AES mode. In DES/TDES mode refer to *Section 49.4.7* to complete the last block.
2. Configure the DMA controller so as to transfer the data to process from the memory to the CRYP peripheral input and the processed data from the CRYP peripheral output to the memory, as described in *Section 49.6: CRYP DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For AES GCM payload encryption or AES CCM payload decryption, the DMA transfer **must not** include the last block. The sequence described in *Appending data using the CPU in polling mode*

must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for CRYP to compute a correct tag.

3. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).
4. Enable DMA requests, by setting DIEN and DOEN of the CRYP_DMACR register.
5. Upon DMA interrupt indicating the transfer completion, get the CRYP-processed data from the memory. Both FIFOs are normally empty and BUSY flag is cleared.

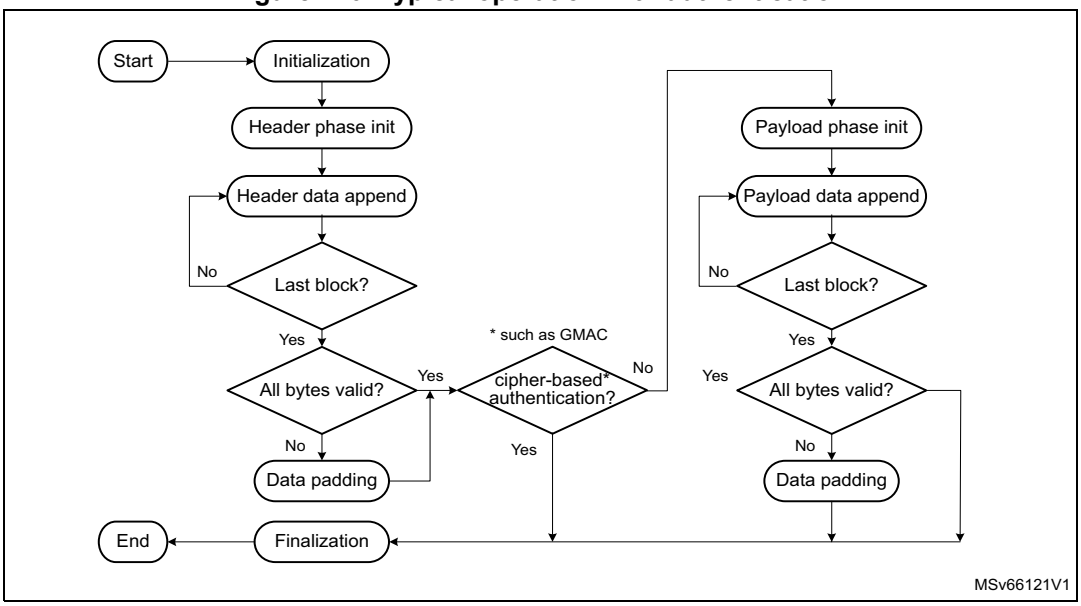
Note: The NPBLB[3:0] bitfield is not used in the header phase of AES GCM, GMAC and CCM chaining modes.

Caution: It is important that the DMA controller empties the CRYP output FIFO before filling up the CRYP input FIFO. To achieve this, the DMA controller must be configured so that the transfer from the CRYP to the memory has a higher priority than the transfer from the memory to the CRYP.

49.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication. This is not applicable to DES/TDES algorithm.

Figure 420. Typical operation with authentication



Section 49.4.12: CRYP AES Galois/counter mode (GCM) and *Section 49.4.14: CRYP AES Counter with CBC-MAC (CCM)* describe detailed sequences supported by CRYP.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by CRYP is described in *Section 49.4.13: CRYP AES Galois message authentication code (GMAC)*.

49.4.7 CRYP ciphertext stealing and data padding

When using DES/TDES or AES algorithm in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (8 bytes for DES/TDES, 16 bytes for AES), the application must use ciphertext stealing techniques such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since CRYP does not implement such techniques, the application must complete the last block of input data using data from the second last block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when the AES algorithm is used in other modes than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since CRYP does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

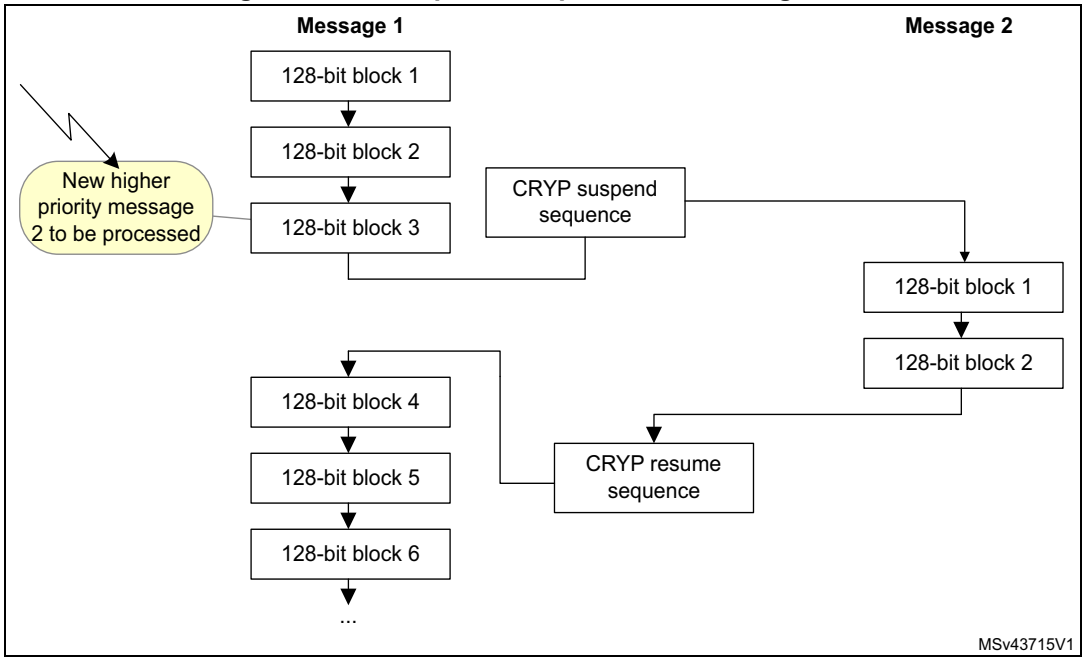
49.4.8 CRYP suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as CRYP is enabled again, to receive a next data block.

Figure 421 gives an example of suspend and resume operation: message 1 is suspended in order to send a shorter and higher-priority message 2 (AES algorithm).

Figure 421. Example of suspend mode management



Detailed descriptions of suspend and resume operations are provided in the sections dedicated to each chaining mode.

49.4.9 CRYP DES/TDES basic chaining modes (ECB, CBC)

Figure 422 and *Figure 423* describe the TDES electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x0. For DES write ALGOMODE[3:0] with 0x2 (only K1 is used).

Figure 422. DES/TDES ECB encryption

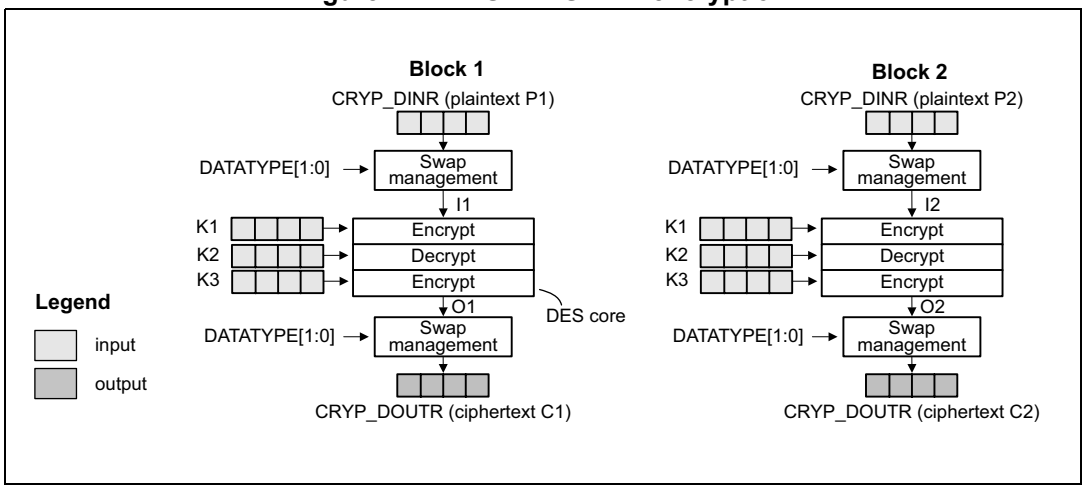


Figure 423. DES/TDES ECB decryption

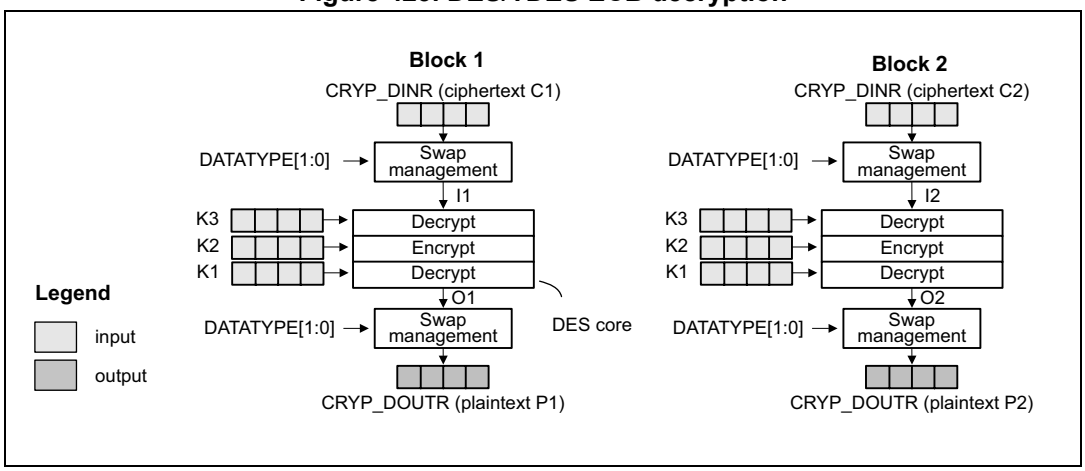


Figure 424 and *Figure 424* describe the TDES cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x1. For DES write ALGOMODE[3:0] with 0x3 (only K1 is used).

Figure 424. DES/TDES CBC encryption

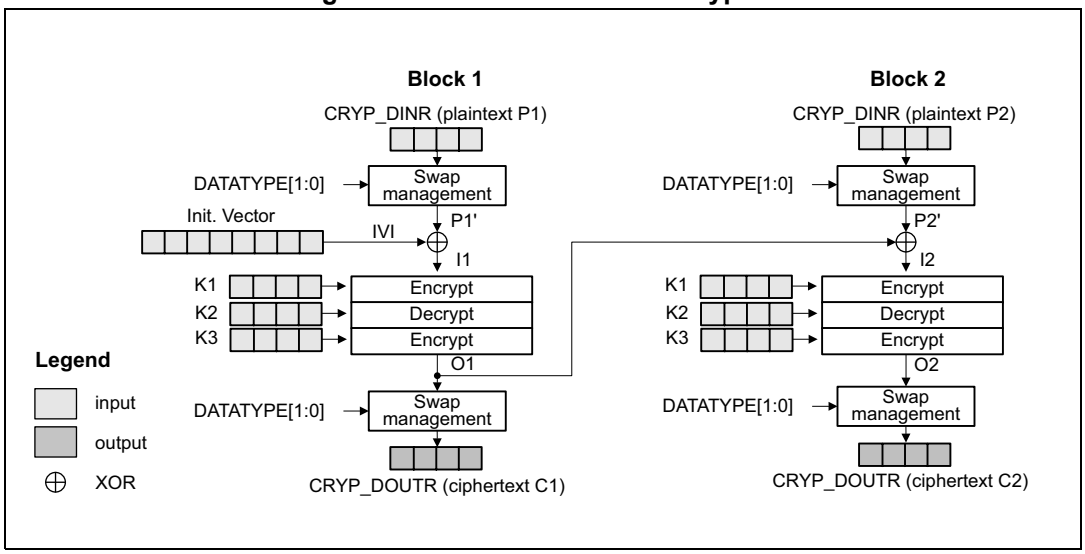
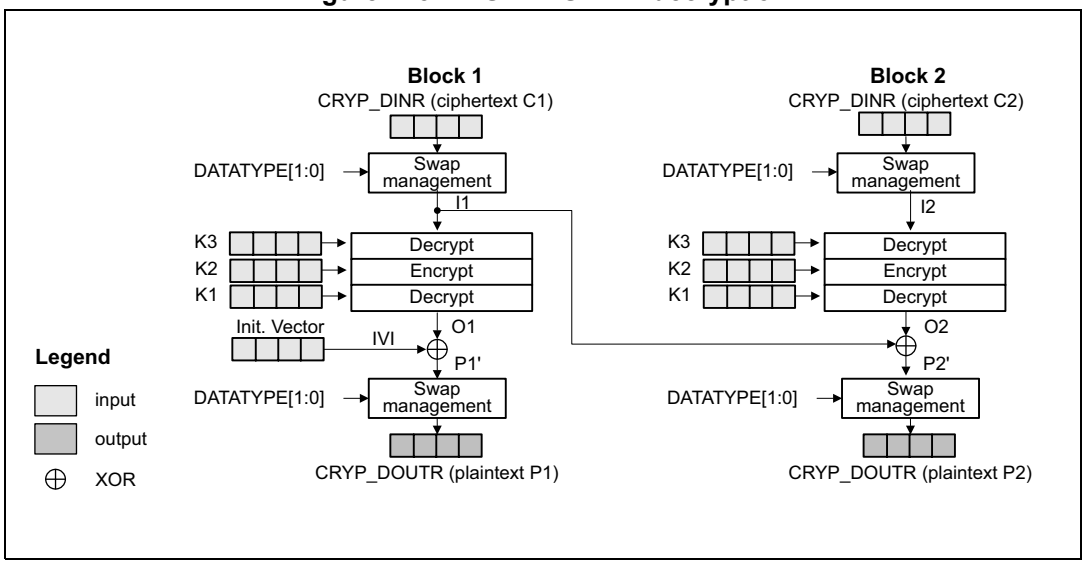


Figure 425. DES/TDES CBC decryption



For more details refer to FIPS PUB 46-3 – 1999 and ANSI X9.52 – 1998.

DES/TDES ECB/CBC encryption and decryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) For DES: select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x2 or 0x3) in encryption or decryption mode (write ALGODIR with 0 or 1).
   For TDES: select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x0 or 0x1) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Normal key mode must be selected (KMOD[1:0] cleared).
4. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
5. Write the key(s) into the CRYP_KxL/R registers.
6. Enable CRYP by setting CRYPEN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 49.4.7*.
   b) Append the cleartext block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the ciphertext block(s), MSB first.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Suspend and resume operations in DES/TDES ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO by clearing the DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared. Alternatively, as the input FIFO can contain up to four unprocessed DES/TDES blocks, the application can decide for real-time reason to interrupt the CRYP without waiting for the input FIFO to be empty. In this case, the alternative is the following:
   a) Wait until output FIFO is empty (OFNE = 0).
   b) Read back the data loaded in the input FIFO that have not been processed and save them in the memory until the input FIFO is empty.
3. If DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP peripheral, by clearing CRYPEN.
5. Save the CRYP_CR register and clear the key registers if they are not needed, to process the higher-priority message.
6. Save initialization vector registers (only required in CBC mode as the CRYP_IVxL/R registers are altered during the data processing).

7. If DMA is used, save the DMA controller status (pointers for CRYP input and output data transfers, number of remaining bytes, and so on).

Note: Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the CRYP_CR register (with correct KEYSIZE), then restore the CRYP_KxL/R registers.
4. Restore the CRYP_IVxL/R registers, using the saved configuration (only required in CBC mode).
5. Optionally, write the data that were saved during context saving into the input FIFO.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA transfers, by setting DIEN and DOEN.

49.4.10 CRYP AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB mode, a special key scheduling is required before processing the first block.

Figure 426 and *Figure 427* describe the AES electronic codebook (ECB) chaining implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x4.

Figure 426. AES-ECB encryption

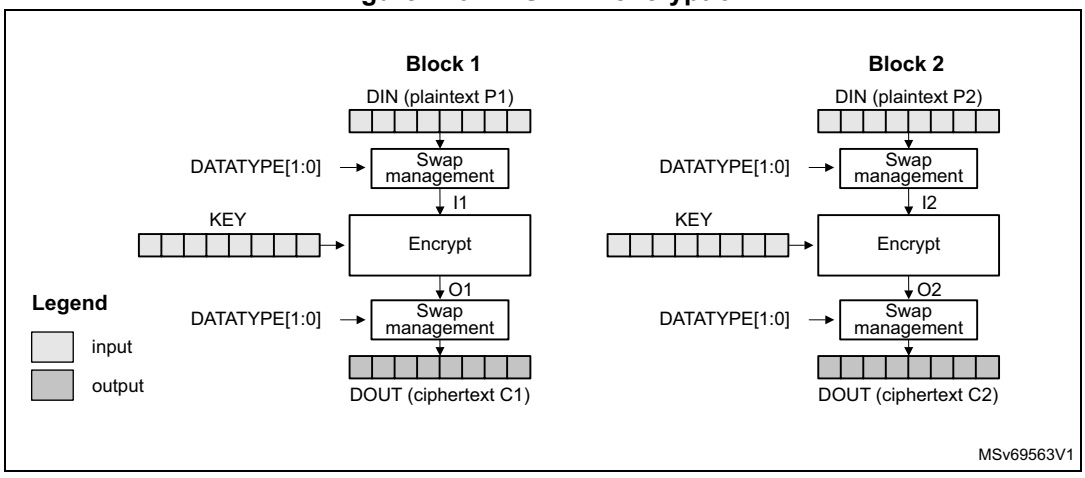
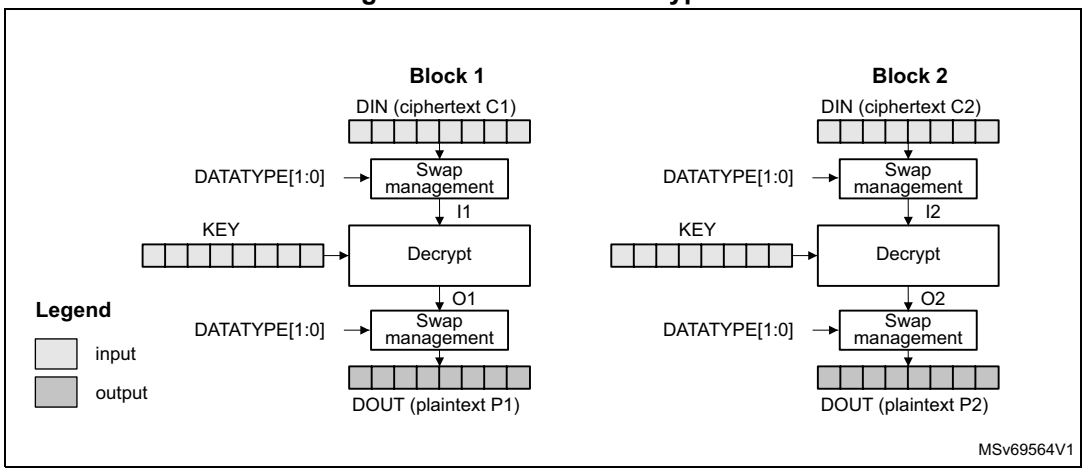


Figure 427. AES-ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC mode, a special key scheduling is required before processing the first block.

Figure 428 and *Figure 429* describe the AES cipher block chaining (CBC) implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x5.

Figure 428. AES-CBC encryption

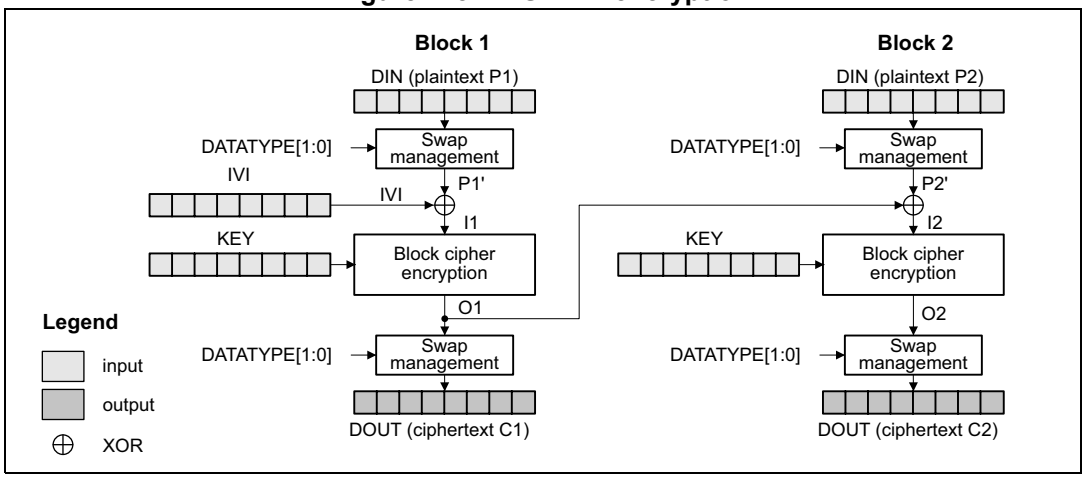
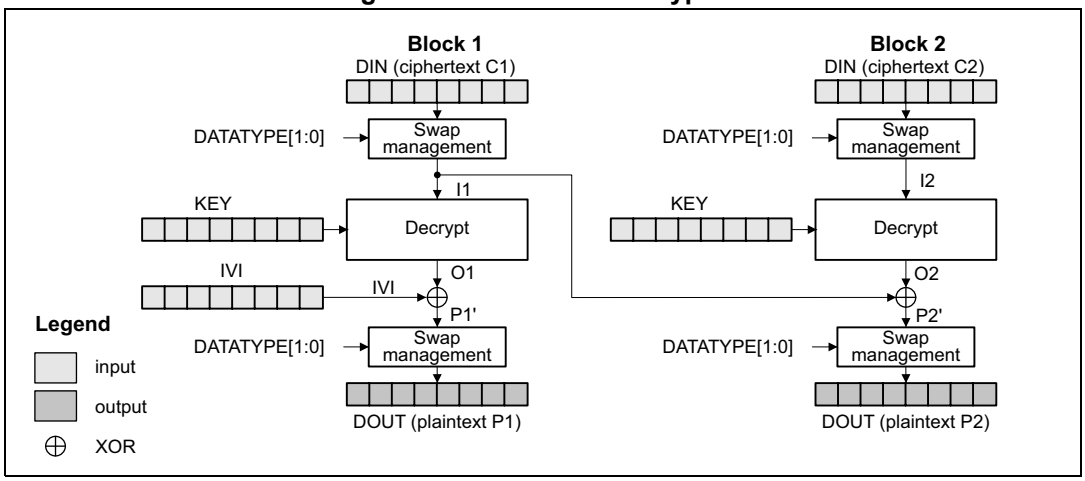


Figure 429. AES-CBC decryption



For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

AES ECB/CBC encryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in encryption mode (ALGODIR = 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode using KMOD[1:0]: if the key is coming from SAES peripheral, set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0] = 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.15*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 49.4.7*.
   b) Append the cleartext block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the ciphertext block(s), MSB first.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time, otherwise KERF is raised if DES or TDES is selected (for example ALGOMODE[3:0] = 0x0).

AES ECB/CBC decryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select the key derivation mode (write ALGOMODE[3:0] with 0x7) in encryption mode (ALGODIR= 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.15*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the CRYP peripheral by setting CRYPEN. The peripheral immediately starts an AES round for key preparation, setting the BUSY bit.
7. Wait until the BUSY bit is cleared, then select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in decryption mode (ALGODIR = 1). Do not change other parameters.
8. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
9. Enable CRYP by setting CRYPEN.
10. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 49.4.7*.
    b) Append the ciphertext block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the cleartext block(s), MSB first.
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
11. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time, otherwise KERF is raised if DES or TDES is selected (for example ALGOMODE[3:0] = 0x0).

Suspend and resume operations in AES ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO by clearing the DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.

3. If DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP peripheral, by clearing CRYPEN.
5. Save the CRYP_CR register and clear the key registers if they are not needed, to process the higher-priority message.
6. Save initialization vector registers (only required in CBC mode as the CRYP_IVxL/R registers are altered during the data processing).
7. If DMA is used, save the DMA controller status (pointers for CRYP input and output data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the CRYP_CR register (with correct KEYSIZE), then restore the CRYP_KxL/R registers. If KMOD[1:0] = 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.15*).
4. Prepare the decryption key as described in *AES ECB/CBC decryption process* (only required for ECB or CBC decryption).
5. Restore the CRYP_IVxL/R registers, using the saved configuration (only required in CBC mode).
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA transfers, by setting DIEN and DOEN.

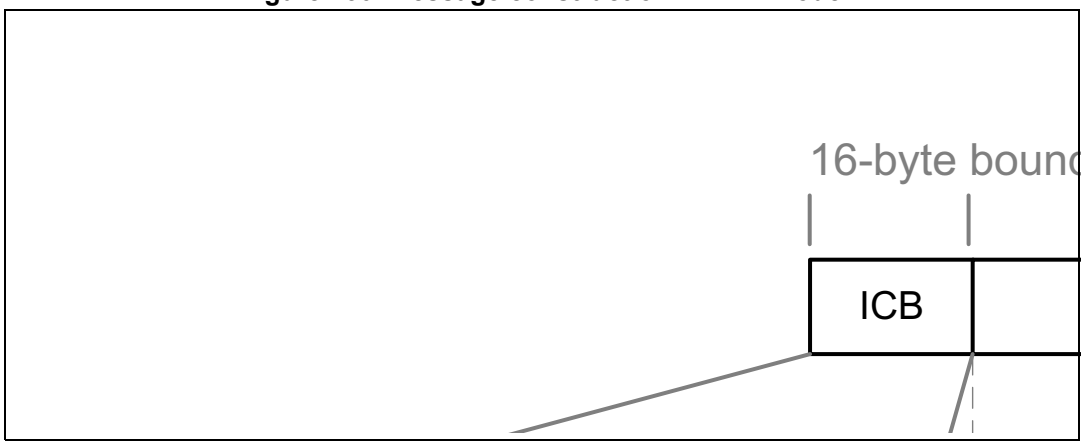
Note: Key registers do not need to be saved as the original key value is known by the application.

49.4.11 CRYP counter mode (AES-CTR)

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 430*.

Figure 430. Message construction in CTR mode



The structure of this message is:

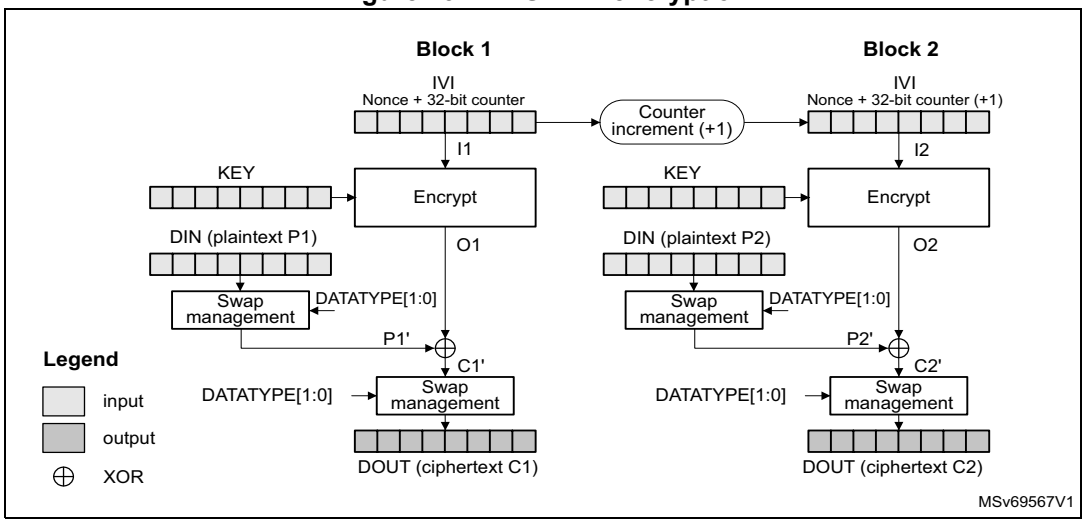
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IVI): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is complete. The initial value of the counter must be set.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 431 describes the counter (CTR) chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 431. AES-CTR encryption



Initialization vectors in CRYP must be initialized as shown in *Table 456*.

Table 456. Counter mode initialization vector definition

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x1

CTR encryption and decryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select CTR chaining mode (write ALGOMODE[3:0] with 0x6) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers according to *Table 456*.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see *Section 49.4.15*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s), MSB first.
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time, otherwise KERF is raised if DES or TDES is selected (for example ALGOMODE[3:0] = 0x0).

CTR mode with DIN = 0x0 (STS mode)

When the application needs to use CTR mode with DIN always equal to zero, write ALGOMODE[3:0] with 0xE, to select the STS mode.

Note: This mode is not documented by NIST.

Suspend and resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 49.4.10: CRYP AES basic chaining modes (ECB, CBC)*.

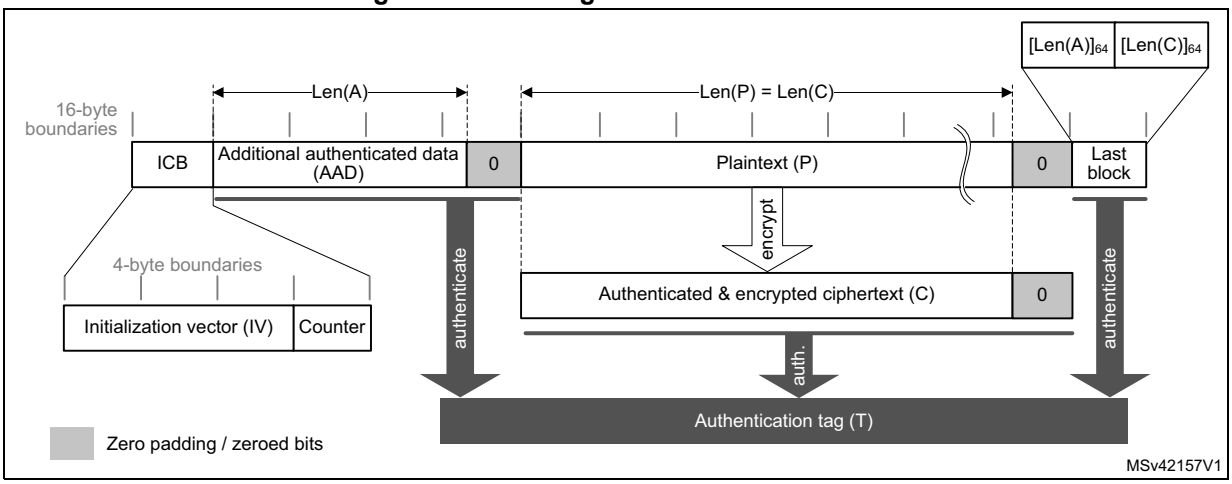
Note: Like for CBC mode, IVI registers must be reloaded during the resume operation.

49.4.12 CRYP AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) enables encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 432. Message construction in GCM



The structure of this message is defined as below:

- **16-byte Initial Counter Block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IVI)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVIs with less than 96 bits, but in this case strict rules apply.
  - **Counter***:* a 32-bit big-endian integer that is incremented each time a block processing is complete. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-bytes blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 458*.

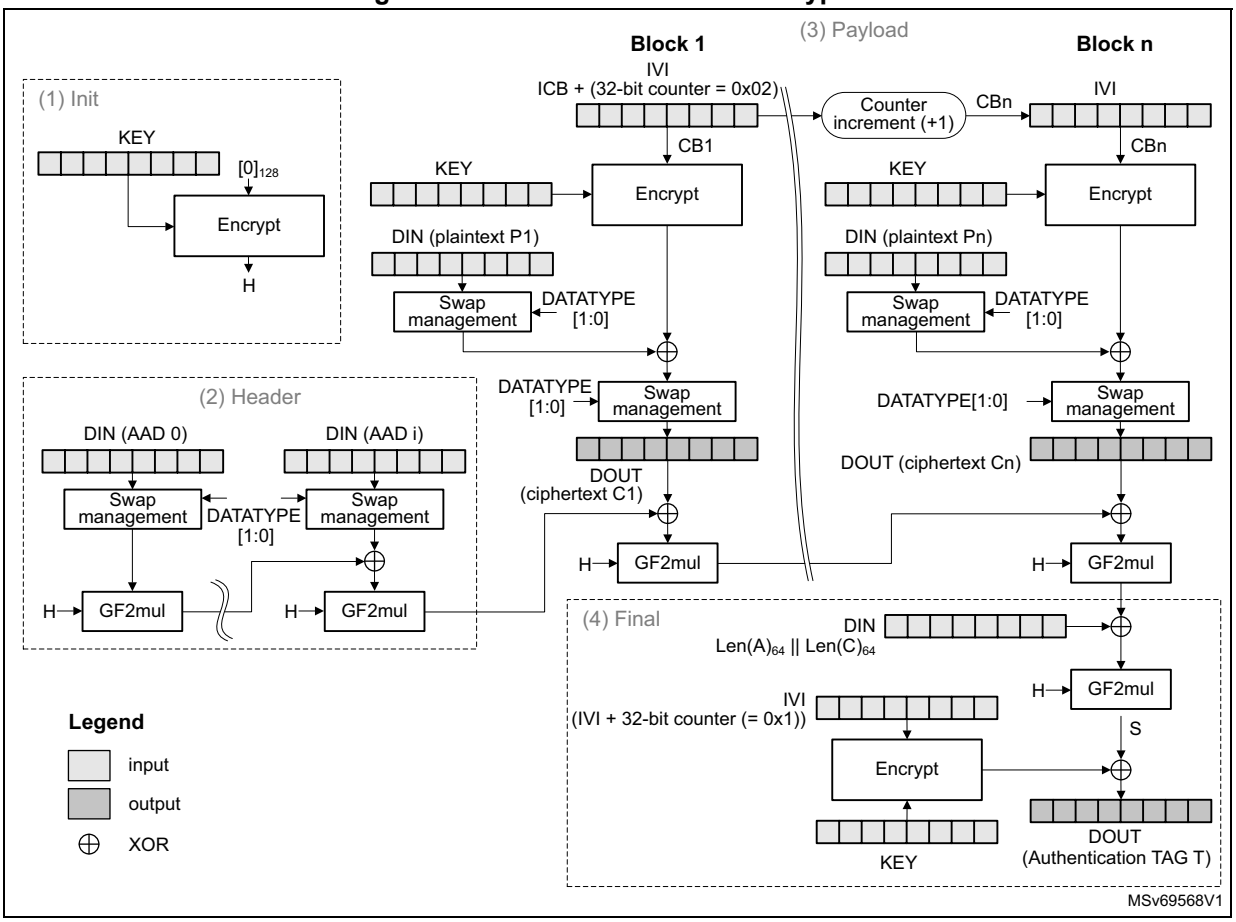
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

A special padding scheme is required when a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes.

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 433 describes the GCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 433. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 457*.

Table 457. GCM mode IVI registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x2

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 458*.

Table 458. GCM last block definition

Word order to CRYP_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 49.4.6*, with the following sequence of events:

GCM initialize

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select GCM chaining mode (write ALGOMODE[3:0] with 0x8) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the GCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector in CRYP_IVxL/R registers according to *Table 457*.
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.15*).
6. Wait until KEYVALID is set (the key loading completed).
7. Set CRYPEN to start the calculation of the hash key. CRYPEN is automatically cleared when the calculation is complete.
8. Wait for the CRYPEN bit to be cleared, indicating that the GCM hash subkey (H) computation is complete.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCM_CCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block follow the steps *a)* and *b)*.
11. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Disable the CRYP peripheral, by clearing CRYPEN.
    b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s).
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.
14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(C) = 0 (see GMAC mode).

GCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the CRYP_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 458*.
17. Wait until the OFNE flag in the CRYP_SR register is set.
18. Get the GCM authentication tag, by reading the CRYP_DOUTR register four times.
19. Disable the CRYP peripheral, by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to CRYP_DINR normally (no swapping).

Suspend and resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers in memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

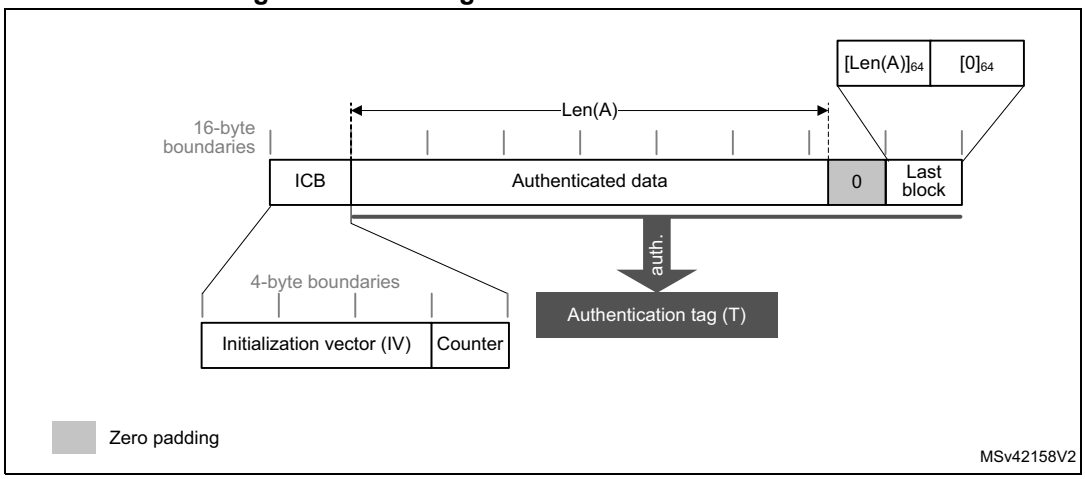
1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.15*).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).

49.4.13 CRYP AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) enables the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

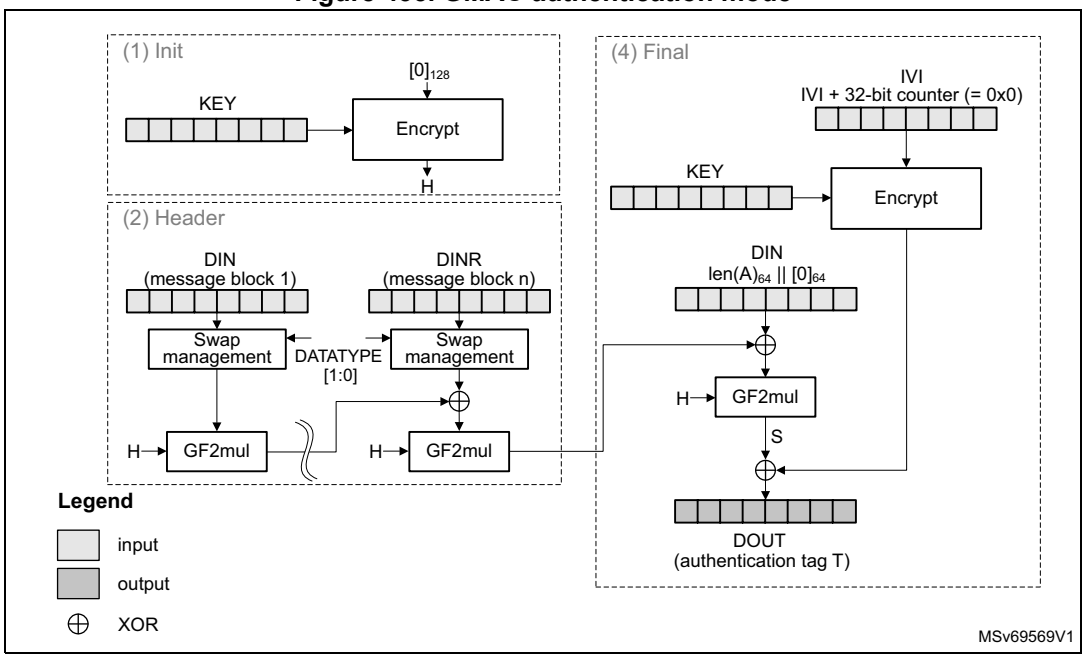
Figure 434. Message construction in GMAC mode



For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 435 describes the GMAC chaining implementation in the CRYP peripheral. To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 435. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as for the GCM, except that the payload phase is omitted.

Suspend and resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

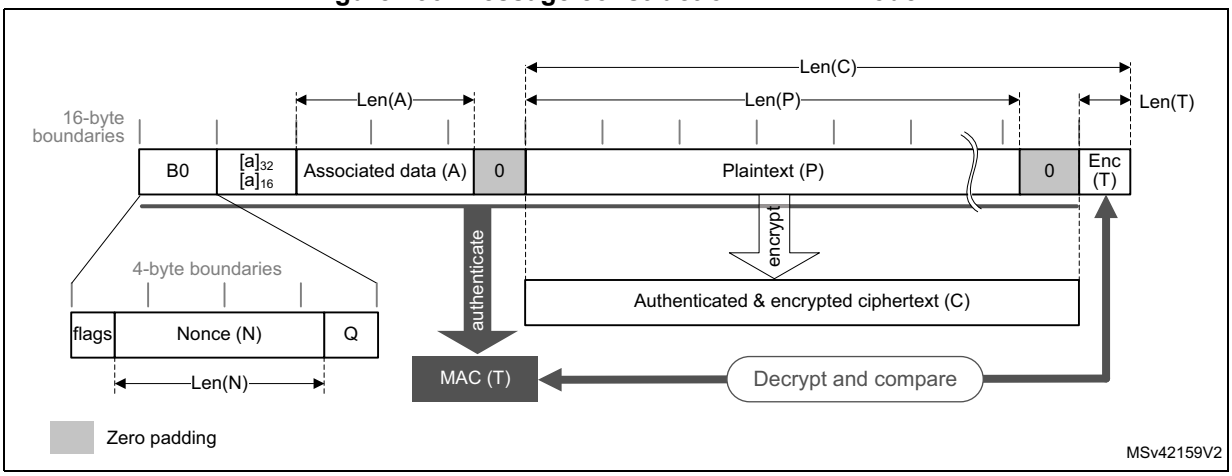
49.4.14 CRYP AES Counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm enables encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 436. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.

- **16-bytes blocks (B)** associated to the Associated Data (A).

  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 436*). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (*a*) must be encoded as follows:

  – If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.

  – If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xFF || 0xFE || $[a]_{32}$, that is, on six bytes.

  – If $2^{32} < a < 2^{64}$, then it is encoded as 0xFF || 0xFF || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 436*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in *NIST Special Publication 800-38C* gives the following values (hexadecimal numbers):

N: **10111213 141516** (Len(N) = 56 bits or 0x7 bytes)

A: **00010203 04050607** (Len(A) = 64 bits or 0x8 bytes)

P: *20212223* (Len(P) = 32 bits i.e. Q = 0x*4* bytes)

T: 6084341b (Len(T) = 32 bits or t = 4)

B0: 4f**101112 13141516** 00000000 00000004
B1: 0008**0001 02030405 0607**0000 00000000
B2: *20212223* 00000000 00000000 00000000

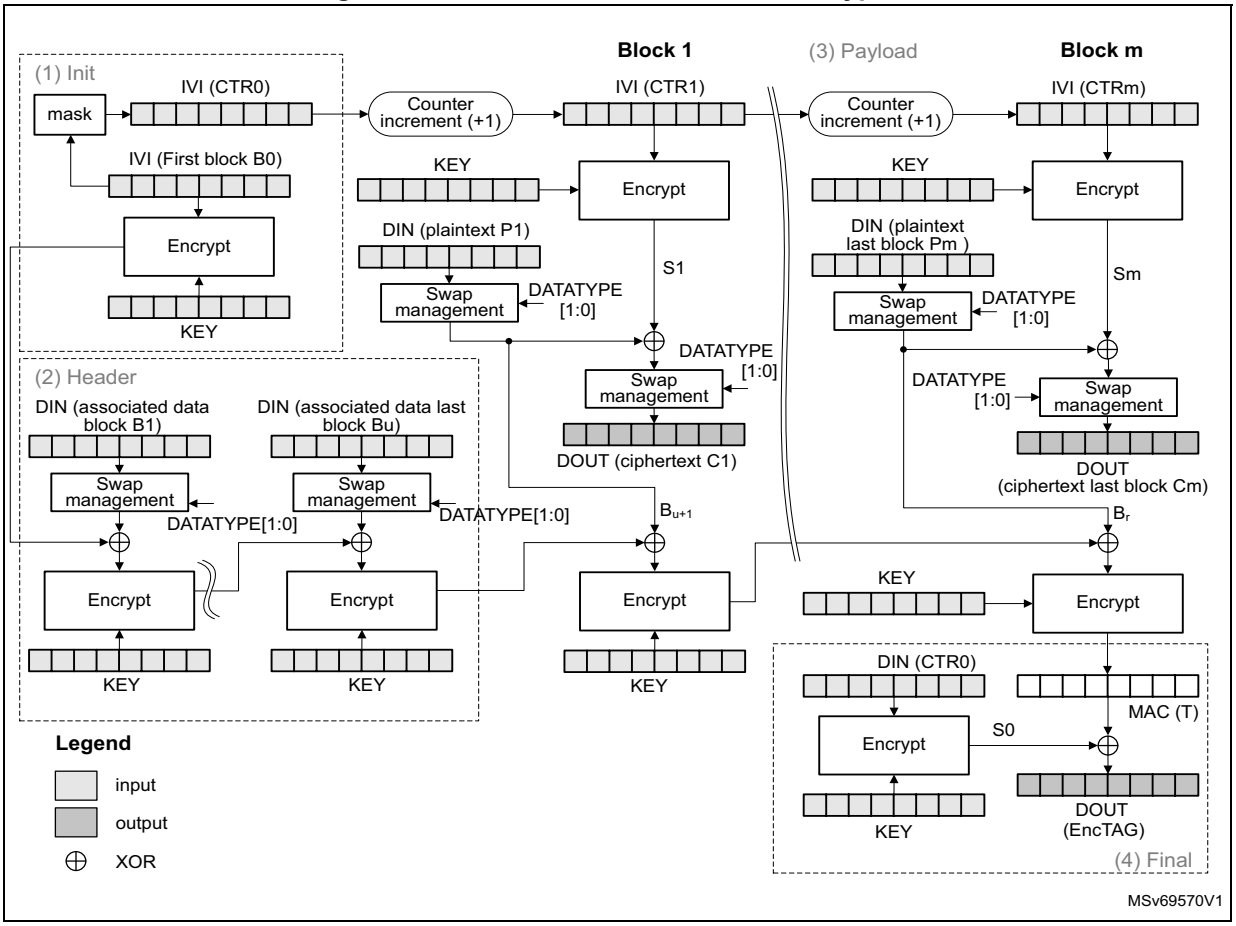
CTR0: 07**10111213 141516** 00000000 00000000

CTR1: 07**10111213 141516** 00000000 00000001

For more details, refer to NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 437 describes the CCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x9.

Figure 437. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 459*.

Table 459. CCM mode IVI registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The 5 most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for bit 0 that is set.

CRYP supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 49.4.6*, with the following sequence of events:

CCM initialize

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select CCM chaining mode (write ALGOMODE[3:0] with 0x9) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the CCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector into the CRYP_IVxL/R registers with CTR1 information, as defined in *Table 459*
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see *Section 49.4.15*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the CRYP peripheral by setting CRYPEN.
8. Write the B0 data in CRYP_DINR registers according to *Table 459*, then wait for the CRYPEN bit to be cleared.

Note: In this initialization phase, data have to be inserted normally (no swapping).

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the GCM header phase, by writing GCM_CCMPH[1:0] with 0x1. Do not change the other configurations written during the GCM initialization.
   c) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block follow the steps *a)* and *b)*.
11. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Disable the CRYP peripheral, by clearing CRYPEN.
    b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s).
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.
14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.
16. Write the CTR0 information into the CRYP_DINR register. CTR0 information is described in *Table 459*, with the bit[0] cleared.
17. Wait until the OFNE flag in the CRYP_SR register is set.
18. Get the CCM authentication tag, by reading the CRYP_DOUTR register four times.
19. Disable the CRYP peripheral by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In this final phase, data have to be inserted normally (no swapping).

Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx registers in the memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.15*).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).

49.4.15 AES key sharing with secure AES co-processor

The CRYP peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral. When it is ready, the CRYP application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in section *SAES operations with shared keys* in section *Secure AES coprocessor (SAES)* of this document.

1. In SAES peripheral, the application encrypts the key to share in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].
2. Each time the shared key is required in the CRYP peripheral, the application decrypts it in the SAES peripheral in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].
3. Once the shared key is decrypted and loaded in SAES_KEYRx registers it can be shared with CRYP (only in AES mode). To load the shared key in CRYP, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the CRYP_KxL/R registers and the BUSY flag in the CRYP_SR register set.
4. Once the key transfer is complete, the BUSY flag is cleared and the KEYVALID flag set in the CRYP_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KERF flag is set in the CRYP_SR register, it means that an AES algorithm has not been selected with ALGOMODE[3:0], the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID in SAES_SR cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is complete, the CRYP is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in CRYP peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

When sharing keys KEYSIZE in CRYP must be identical to SAES value (128 or 256 bits).

Caution: Avoid changing KEYSIZE or ALGOMODE[3:0] to a DES or TDES algorithm when the key registers have been initialized with a shared key. This causes key data to be lost and the KERF error to be raised.

49.4.16 CRYP data registers and data swapping

DES/TDES data input and output

A 8-byte data block enters the CRYP peripheral with two successive 32-bit word writes into the CRYP_DINR register (bitfield DIN[31:0]), the most significant word (bits [64:33]) first, the least significant word (bits [32:1]) last.

A 8-byte data block is retrieved from the CRYP peripheral with two successive 32-bit word reads of the CRYP_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [64:33]) first, the least significant word (bits [32:1]) last.

DES/TDES data swapping feature

The CRYP peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the CRYP input FIFO, before loading it to the DES processing core, and on the data output from the DES processing core, before sending it to the CRYP output FIFO. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0]. The selection applies both to the input and the output of the CRYP.

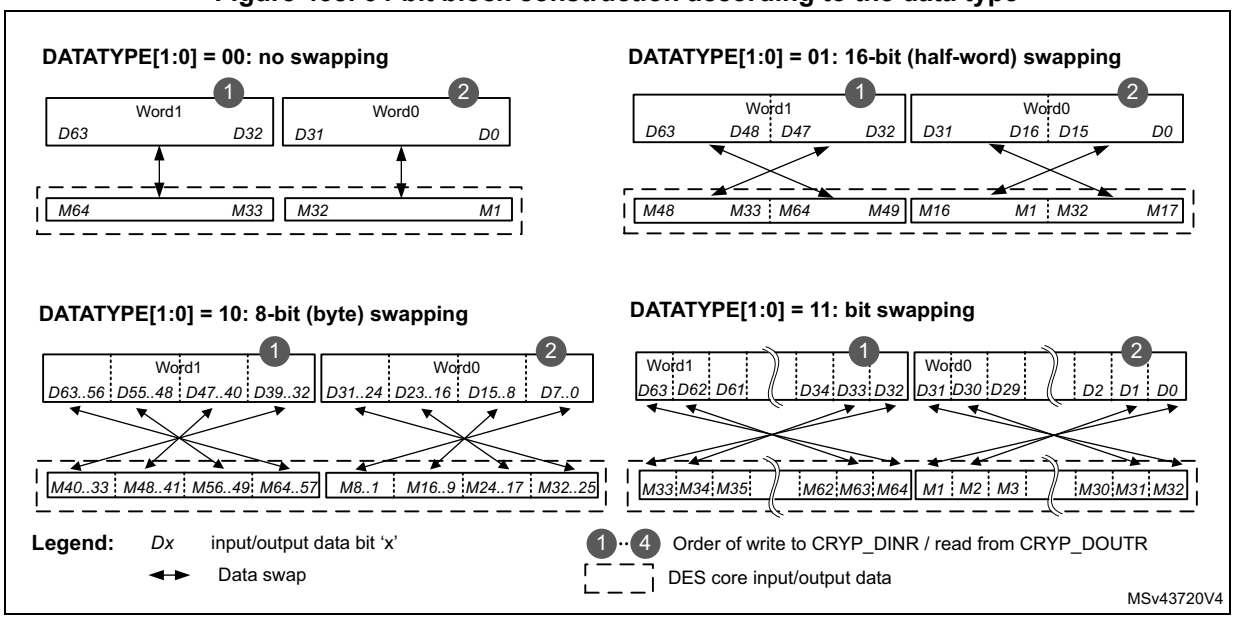
Note: The data in CRYP key registers (CRYP_KxL/R) and initialization vector registers (CRYP_IVxL/R) are not sensitive to the swap mode selection.

The CRYP data swapping feature for DES/TDES is summarized in *Table 460* and *Figure 438*.

Table 460. DES/TDES data swapping example

DATATYPE[1:0]	Swapping performed	Data block representation (64 bits)
		System memory data (big-endian)
0x0	No swapping	Block[64..1]: 0xABCD7720 6973FE01
		Address @, word[63..32]: 0xABCD 7720 Address @+0x4, word[31..0]: 0x6973 FE01
0x1	Half-word (16-bit) swapping	Block[64..1]: 0x**ABCD** 7720 **6973** FE01
		Address @, word[63..32]: 0x7720 **ABCD** Address @+0x4, word[31..0]: 0xFE01 **6973**
0x2	Byte (8-bit) swapping	Block[64..1]: 0x**AB** CD **77** 20 **69** 73 **FE** 01
		Address @, word[63..32]: 0x20**77** CD**AB** Address @+0x4, word[31..0]: 0x01**FE** 73**69**
0x3	Bit swapping	Block[64..33]: 0xABCD 7720 1010 1011 1100 1101 0111 0111 0010 0000 Block[32..1]: 0x6973 FE01 0110 1001 0111 0011 1111 1110 0000 0001
		Address @, word[63..32]: 0x04EE B3D5 0000 0100 1110 1110 1011 0011 1101 0101 Address @+4, word[31..0]: 0x807F CE96 1000 0000 0111 1111 1100 1110 1001 0110

Figure 438. 64-bit block construction according to the data type



AES data input and output

A 16-byte data block enters the CRYP peripheral with four successive 32-bit word writes into the CRYP_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the CRYP peripheral with four successive 32-bit word reads of the CRYP_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 461* “no swapping” option for details.

AES data swapping feature

The CRYP peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the CRYP input FIFO, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the CRYP output FIFO. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0]. The selection applies both to the input and the output of the CRYP, except in GCM/CCM final phase (data must be inserted normally).

Note: The data in CRYP key registers (CRYP_KxL/R) and initialization vector registers (CRYP_IVxL/R) are not sensitive to the swap mode selection.

The CRYP data swapping feature for AES is summarized in *Table 461* and *Figure 439*.

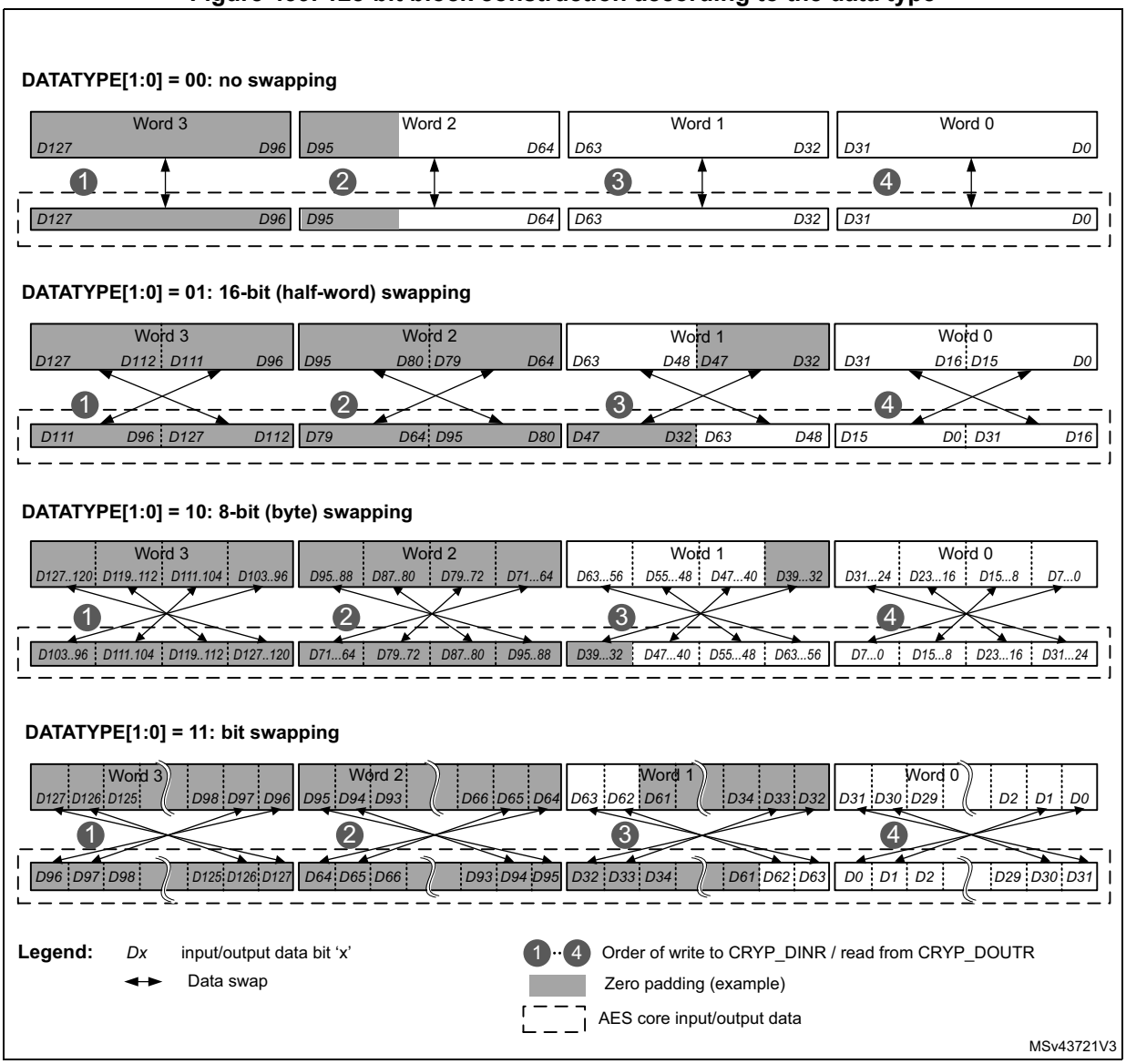
Table 461. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EE F672 Address @ + 0x4, word[95..64]: 0x2E04 CE96 Address @ + 0x8, word[63..32]: 0x4E6F 7720 Address @ + 0xC, word[31..0]: 0x6973 2074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

Table 461. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x3	Bit swapping	Block[63..32]: 0x4E6F 7720 **0100 1110 0110 1111 0111 0111 0010 0000** Block[31..0]: 0x6973 2074 **0110 1001 0111 0011 0010 0000 0111 0100**
		Address @, word[63..32]: 0x04EE F672 **0000 0100 1110 1110 1111 0110 0111 0010** Address @ + 0x4, word[31..0]: 0x2E04 CE96 **0010 1110 0000 0100 1100 1110 1001 0110**

Figure 439. 128-bit block construction according to the data type



Data padding

Figure 439 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

49.4.17 CRYP key registers

The eight CRYP_KxR/LR write-only registers store the encryption or decryption key information, as shown in *Table 462* and *Table 463*. Reads are not allowed for security reason.

When TDES is selected a 3-key vector (K1, K2, and K3) is used. When DES is selected only 1-key vector (K1) is used.

Note: In memory and in CRYP *key registers, keys are stored in big-endian format, with most significant byte on the lowest address.*

Table 462. Key endianness in CRYP_KxR/LR registers (AES 128/192/256-bit keys)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	-	-	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
k[255:224]	k[223:192]	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]

Table 463. Key endianness in CRYP_KxR/LR registers (DES K1 and TDES K1/2/3)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	K1[64:33]	K1[32:1]	-	-	-	-
-	-	K1[64:33]	K1[32:1]	K2[64:33]	K2[32:1]	K3[64:33]	K3[32:1]

Note: DES/TDES keys include 8-bit parity information that are not used by the cryptographic processor. In other words, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value Kx[1:64] are not used.

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the CRYP_KxL/R registers are ignored when BUSY bit is set. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes (in ascending order):

- K2LR, K2RR, K3LR, then K3RR registers for KEYSIZE = 0x0
- K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x1
- K0LR, K0RR, K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x2

Similarly, the entire key(s) must be written before starting a DES or TDES computation. The key registers must always be written in either ascending or descending order. The write sequence becomes (in ascending order):

- K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for ALGOMODE[3:0] = 0x0 or 0x1
- K1LR then K1RR for ALGOMODE[3:0] = 0x2 or 0x3

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key.

As soon as the first key register is written, the KEYVALID bit is cleared. Once the key registers writing sequence is complete, KEYVALID is set and CRYPEN bit becomes writable. If an error occurs, KEYVALID is cleared and KERF set (see *Section 49.4.19*).

49.4.18 CRYP initialization vector registers

The four CRYP_IVxR/LR registers store the initialization vector (IVI) information, as shown in *Table 464* and *Table 465*. They can only be written if BUSY bit is cleared.

Note: In memory and in CRYP IVI registers, initialization vectors *are stored in big-endian format, with most significant byte on the lowest address.*

Table 464. Initialization vector endianness in CRYP_IVxR registers (AES)

CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Table 465. Initialization vector endianness in CRYP_IVxR registers (DES/TDES)

CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
IVI[64:32]	IVI[31:0]	-	-

Initialization vector information depend on the chaining mode selected. When used, CRYP_IVxR/LR registers are updated upon each DES/TDES or AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled by DATATYPE[1:0].

49.4.19 CRYP error management

The CRYP peripheral manages the errors described in this section.

Key error flag (KERF)

There are multiple sources of errors that set the KERF flag and clear the KEYVALID bit of the CRYP_SR register:

- **Key writing sequence error**: triggered upon detecting an incorrect sequence of writing key registers. See *Section 49.4.17: CRYP key registers* for details.
- **Key sharing size mismatch error**: triggered when KMOD[1:0] is at 0x2 and KEYSIZE in CRYP does not match KEYSIZE in SAES peripheral, or KEYSIZE is at 0x1 in CRYP_CR (192-bit key).
- **Key sharing error**: triggered upon failing transfer of SAES shared key to CRYP. See *Section 49.4.15: AES key sharing with secure AES co-processor* for details.
- **DES/TDES selection while in key sharing mode**: DES or TDES is selected with ALGOMODE[3:0], while KMOD[1:0] is 0x2 (shared-key mode).

Upon a key writing sequence error, KERF flag is cleared when a correct key writing sequence starts.

Upon a key sharing error, reset both CRYP and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KERF flag prior to setting CRYPEN.

49.5 CRYP interrupts

There are two individual maskable interrupt sources generated by the CRYP peripheral to signal the following events:

- Input FIFO empty (IFEM) or not full (IFNF)
- Output FIFO full (OFFU) or not empty (OFNE)

These sources are combined into a common interrupt signal from the CRYP peripheral that connects to the Arm® Cortex® interrupt controller. The application can enable or disable CRYP interrupt sources individually by setting/clearing the corresponding mask bit of the CRYP_IMSCR register. The status of the individual source of event flags can be read from the CRYP_SR register.

The global status of the two source of interrupts can be read either from the CRYP_RISR register (raw, that is unmasked), or from the CRYP_MISR register (masked).

Table 466 gives a summary of the available features.

Table 466. CRYP interrupt requests

Interrupt acronym	Interrupt event	Event flag		Enable bit	Interrupt clear method
		in CRYP_xISR(1)	in CRYP_SR		
CRYP	Output FIFO full	OUTRIS (not masked) OUTMIS (masked)	OFFU	OUTIM(2) and CRYPEN	Read one data from output FIFO
	Output FIFO not empty		OFNE		Read all data from output FIFO
	Input FIFO not full	INRIS (not masked) INMIS (masked)	IFNF	INIM(2) and CRYPEN	Write data until input FIFO is full
	Input FIFO empty		IFEM		Write at least one word in input FIFO

1. Flag of the CRYP_RISR register (unmasked or raw), or CRYP_MISR register (masked).
2. Bit of the CRYP_IMSCR register.

Output FIFO service interrupt (OUTMIS)

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is, when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is active also when CRYP is disabled. Consequently, when CRYPEN bit is cleared, the OUTMIS bit is not cleared when the output FIFO is not empty.

Input FIFO service interrupt (INMIS)

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is active only when CRYP is enabled. Consequently, when CRYPEN bit is cleared, the INMIS bit stay low even if the input FIFO is empty.

49.6 CRYP DMA requests

The CRYP peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DIEN and DOEN bits of the CRYP_DMACR register.

Detailed usage of DMA with CRYP can be found in *Appending data using the DMA* subsection of *Section 49.4.5: CRYP encryption/ decryption typical usage*.

Data input using DMA

Setting DIEN enables DMA writing into CRYP. The CRYP then initiates, during the input phase, a DMA request for each 16-byte data block to write to the CRYP_DINR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration in AES mode is fixed burst size of four words. When reading data from DDR-SDRAM memory, it is recommended to use 16-word burst transfers for better performance.

In AES mode CRYP issues a DMA transfer request to its input FIFO every time a 16-bytes block has been read from the FIFO. In DES/TDES mode it corresponds to two 8-byte blocks.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 49.4.7).

Data output using DMA

Setting DOEN enables DMA reading from CRYP. The CRYP then initiates, during the output phase, a DMA request for each 16-byte data block to read from the CRYP_DOUTR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration in AES mode is fixed burst size of four words. When writing data to DDR-SDRAM memory, it is recommended to use 16-word burst transfers for better performance. When DES or TDES is selected, single DMA transfers must be used.

In AES mode CRYP issues a DMA transfer request from output FIFO every time one 16-byte block has been written into the FIFO. In DES/TDES mode CRYP issues a DMA transfer request every time a 32-bit word has been written into the FIFO.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when CRYP is disabled (CRYPEN cleared) or the DMA enable bit (DIEN for input data, DOEN for output data) is cleared.

49.7 CRYP processing time

The following tables provide the block processing latency per operating mode. Block size is 16-byte in AES mode, 8-bytes in DES/TDES mode.

Table 467. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
64 bits	DES encryption or decryption	ECD, CBC	16
3x64 bits	TDES encryption or decryption		48
128 bits	AES encryption or decryption(1)	ECD, CBC, CTR	14
	AES key preparation	-	12
192 bits	AES encryption or decryption(1)	ECD, CBC, CTR	16
	AES key preparation	-	14

Table 467. Processing latency for ECB, CBC and CTR (continued)

256 bits	AES encryption or decryption(1)	ECD, CBC, CTR	18
	AES key preparation	-	16

1. Excluding key preparation time (ECB and CBC only).

Table 468. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase	Payload phase	Final phase
128 bits	AES encryption or decryption	GCM	24	10	14	14
		CCM	12	14	25	14
192 bits		GCM	28	10	16	16
		CCM	14	16	29	16
256 bits		GCM	32	10	18	18
		CCM	16	18	33	18

49.8 CRYP registers

These registers are accessible through 32-bit word single accesses only (otherwise an AHB error is generated, and write accesses are ignored).

49.8.1 CRYP control register (CRYP_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				ALGOMODE[3]	Res.	GCM_CCMPH[1:0]	
rw						rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **IPRST**: CRYP peripheral software reset

Setting the bit resets the CRYP peripheral, putting all registers to their default values, except the IPRST bit itself.

This bit must be kept cleared while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

This bitfield defines how the CRYP key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal-key mode. Key registers are freely usable.

0x2: Shared-key mode. If shared-key mode is properly initialized in SAES peripheral, the CRYP peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in CRYP key registers when BUSY bit is cleared and KEYVALID is set in the CRYP_SR register. Key error flag KERF is set otherwise in the CRYP_SR register.

Others: Reserved

Shared-key mode can only be used in AES mode.

Attempts to write the bitfield are ignored when BUSY is set.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Attempts to write the bitfield are ignored when BUSY is set.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **GCM_CCMPH[1:0]**: GCM or CCM phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.

0x0: Initialization phase

0x1: Header phase

0x2: Payload phase

0x3: Final phase

Attempts to write the bitfield are ignored when BUSY is set.

Bit 15 **CRYPEN:** CRYP enable

This bit enables/disables the CRYP peripheral.

0:CRYP disabled

1: CRYP enabled

This bit is automatically cleared by hardware upon the completion of the key preparation (ALGOMODE[3:0] at 0x7) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared.

Bit 14 **FFLUSH:** FIFO flush

This bit enables/disables the flushing of CRYP input and output FIFOs. Reading this bit always returns 0.

0: No effect

1: FIFO flush enabled

When CRYPEN is cleared, writing this bit to 1 flushes both input and output FIFOs (that is read and write pointers of the FIFOs are reset).

FFLUSH bit must be set when BUSY is cleared, otherwise the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.

Attempts to write FFLUSH are ignored when CRYPEN is set.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:8 **KEYSIZE[1:0]**: Key size selection

This bitfield defines the key length in bits of the key used by CRYP in AES mode (that is, it has no effect in the DES or TDES modes).
0x0: 128-bits
0x1: 192 bits
0x2: 256 bits
0x3: Reserved
When KEYSIZE is changed, KEYVALID bit is cleared.
Attempts to write the bitfield are ignored when BUSY is set.

Bits 7:6 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the CRYP_DINR register or read from the CRYP_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 49.4.16: CRYP data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data).
0x2: Byte swapping (8-bit data).
0x3: Bit-level swapping.
Attempts to write the bitfield are ignored when BUSY is set.

Bits 19, 5:3 **ALGOMODE[3:0]:** Algorithm mode

This bitfield selects the algorithm/chaining mode.
0x0: TDES-ECB (triple-DES Electronic Codebook)
0x1: TDES-CBC (triple-DES Cipher Block Chaining)
0x2: DES-ECB (simple DES Electronic Codebook)
0x3: DES-CBC (simple DES Cipher Block Chaining)
0x4: AES electronic codebook (AES-ECB)
0x5: AES cipher Block Chaining (AES-CBC)
0x6: AES counter mode (AES-CTR)
0x7: AES key preparation for ECB or CBC decryption
0x8: AES Galois counter mode (AES-GCM) and Galois message authentication code (AES-GMAC)
0x9: AES counter with CBC-MAC (AES-CCM)
0xE: AES Counter mode with DIN = 0 (AES-STS)
Others: Reserved
Changing ALGOMODE[3:0] from AES algorithm to DES or TDES (or the reverse) clears KEYVALID. KEYVALID is not cleared when the algorithm type is kept between two ALGOMODE[3:0], that is between TDES-ECB and TDES-CBC or between AES-ECB and AES_CTR.
Attempts to write the bitfield are ignored when BUSY is set.

Bit 2 **ALGODIR:** Algorithm direction

This bit selects the algorithm direction.
0: Encryption
1: Decryption
Attempts to write the bitfield are ignored when BUSY is set.

Bits 1:0 Reserved, must be kept at reset value.

49.8.2 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY VALID	KERF	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
								r	r		r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This read-only bit is set by hardware when the key of size defined by KEYSIZE is loaded in CRYP_KxR/LR key registers. It is valid for both AES and DES/TDES algorithms.

0: Key not valid

1: Key valid

The CRYPEN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KERF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by CRYP. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KERF is set, BUSY cleared and KEYVALID remains cleared.

If set, KERF must be cleared, otherwise KEYVALID cannot be set. For further information on key loading, refer to *Section 49.4.17: CRYP key registers*.

Bit 6 **KERF**: Key error flag

This read-only bit is set by hardware when key information failed to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

KERF is triggered upon any of the following errors:

- CRYP_KxR/LR register write does not respect the correct order (refer to *Section 49.4.17: CRYP key registers* for details).
- CRYP fails to load the AES key shared by SAES peripheral (KMOD = 0x2).
- Application wrote KMOD = 0x2 with ALGOMODE[3:0] selecting either DES or TDES algorithm.

KERF must be cleared by the application software, otherwise KEYVALID cannot be set. It can be done through IPRST bit of CRYP_CR, or when a correct key writing sequence starts.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **BUSY:** Busy bit

This flag indicates whether CRYP is idle or busy.

0: Idle

1: Busy

CRYP is flagged as idle when disabled (CRYPEN = 0) or when the AES or DES core is not processing any data. It happens when the last processing has completed, or CRYP is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).

CRYP is flagged as busy when processing a block data, preparing a key (AES ECB or CBC decryption only), or transferring a shared key from SAES peripheral.

Bit 3 **OFFU:** Output FIFO full flag
0: Output FIFO is not full
1: Output FIFO is full

Bit 2 **OFNE:** Output FIFO not empty flag
0: Output FIFO is empty
1: Output FIFO is not empty

Bit 1 **IFNF:** Input FIFO not full flag
0: Input FIFO is full
1: Input FIFO is not full
This flag is not used when STS mode is selected in CRYP_CR register.

Bit 0 **IFEM:** Input FIFO empty flag
0: Input FIFO is not empty
1: Input FIFO is empty
This flag is not used when STS mode is selected in CRYP_CR register.

49.8.3 CRYP data input register (CRYP_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DIN[31:0]:** Data input
In AES mode four-fold sequential write to this bitfield during the Input phase results in pushing a complete 16-byte block into the CRYP input FIFO. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].
In DES/TDES mode a two-fold sequential write to this bitfield during the Input phase results in pushing a complete 8-bytes block into the CRYP input FIFO. From the first to the second write, the corresponding data weights are [63:32] then [31:0].
Input FIFO can receive up to two 16-byte blocks of plaintext (when encrypting) or ciphertext (when decrypting) in AES mode. In DES/TDES mode it is up to four 8-byte blocks.
If EN bit is set in CRYP_CR register, when at least four 32-bit words have been pushed into the input FIFO (two in DES/TDES mode), and when at least four 32-bit words are free in the output FIFO (two in DES/TDES mode), the CRYP automatically starts an encryption or decryption process, setting the BUSY bit.
Reading this register pops data off the input FIFO (oldest value is returned). The data present in the input FIFO are returned only if CRYPEN is cleared (undefined value is returned otherwise). Following one or more reads the FIFO must be flushed (setting the FFLUSH bit) prior to processing new data.

49.8.4 CRYP data output register (CRYP_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]:** Data output

In AES mode four-fold sequential read to this bitfield during the output phase results in retrieving a complete 16-byte block from the CRYP output FIFO. From the first to the fourth read, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].

In DES/TDES mode a two-fold sequential read to this bitfield during the output phase results in retrieving a complete 8-byte block from the CRYP input FIFO. From the first to the second read, the corresponding data weights are [63:32] then [31:0].

Output FIFO can store up to two 16-byte blocks of plaintext (when decrypting) or ciphertext (when encrypting) in AES mode. In DES/TDES mode it is up to four 8-byte blocks.

When the output FIFO is empty a read returns an undefined value. Writes are ignored.

49.8.5 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DOEN:** DMA output enable

When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.

0: Outgoing data transfer from CRYP via DMA is disabled

1: Outgoing data transfer from CRYP via DMA is enabled

Bit 0 **DIEN:** DMA input enable

When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.

0: Incoming data transfer to CRYP via DMA is disabled

1: Incoming data transfer to CRYP via DMA is enabled

49.8.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTIM:** Output FIFO service interrupt mask

This bit enables or disables (masks) the CRYP output FIFO service interrupt generation when OUTRIS is set.

0: Output FIFO interrupt is disabled (masked), masked interrupt status (OUTMIS) stays cleared

1: Output FIFO interrupt is enabled (not masked)

Bit 0 **INIM:** Input FIFO service interrupt mask

This bit enables or disables (masks) the CRYP input FIFO service interrupt generation when INRIS is set.

0: Input FIFO interrupt is disabled (masked), masked interrupt status (INMIS) stays cleared

1: Input FIFO interrupt is enabled (not masked)

49.8.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status

This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register, regardless of the OUTIM mask bit value in CRYP_IMSCR register.

0: No output FIFO event detected

1: Output FIFO full or not empty detected; an interrupt is generated if OUTIM bit is set in CRYP_IMSCR register

Bit 0 **INRIS:** Input FIFO service raw interrupt status

This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register, regardless of the INIM mask bit value in CRYP_IMSCR register.

0: No input FIFO event detected

1: Input FIFO empty or not full detected; an interrupt is generated if CRYPEN is set and if INIM bit is set in CRYP_IMSCR register

49.8.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status

This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register. If the OUTIM mask bit is cleared in CRYP_IMSCR register, the OUTMIS bit stays cleared (masked).

0: No output FIFO event detected or OUTIM mask cleared in CRYP_IMSCR

1: Output FIFO full or not empty detected, with an interrupt pending

The OUTMIS bit is cleared by reading data from the output FIFO until OFNE flag is cleared (output FIFO empty). It is not cleared by disabling CRYP with CRYPEN bit.

Bit 0 **INMIS:** Input FIFO service masked interrupt status

This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register. If the INIM mask bit is cleared in CRYP_IMSCR register, the INMIS bit stays cleared (masked).

0: No input FIFO event detected or INIM mask cleared in CRYP_IMSCR or CRYPEN bit cleared.

1: Input FIFO empty or not full detected, with an interrupt pending

The INMIS bit is cleared by writing data to the input FIFO until IFEM flag is cleared (there is at least one word in input FIFO), or by clearing CRYPEN,

When CRYP is disabled, INMIS bit stays low even if the input FIFO is empty.

49.8.9 CRYP key register 0L (CRYP_K0LR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[255:224]**: Key bit x (x = 255 to 224)

This write-only bitfield contains the bits [255:224] of the AES encryption or decryption key, depending on the operating mode. This register is not used in DES/TDES mode.

Write to CRYP_KxR/LR registers is ignored when CRYP is busy (BUSY bit set). When key is coming from the SAES peripheral (KMOD[1:0] = 0x2) in AES mode, write is also ignored. With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to CRYP_KxR/LR register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the CRYP_SR register.

49.8.10 CRYP key register 0R (CRYP_K0RR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[223:192]**: Key bit x (x = 223 to 192)

This write-only bitfield contains the bits [223:192] of the AES encryption or decryption key, depending on the operating mode. This register is not used in DES/TDES mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.11 CRYP key register 1L (CRYP_K1LR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[191:160]**: Key bit x (x = 191 to 160)

This write-only bitfield contains the bits [191:160] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K1, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[159:128]**: Key bit x (x = 159 to 128)

This write-only bitfield contains the bits [159:128] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K1, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[127:96]**: Key bit x (x = 127 to 96)

This write-only bitfield contains the bits [127:96] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K2, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[95:64]**: Key bit x (x = 95 to 64)

This write-only bitfield contains the bits [95:64] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K2, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[63:32]**: Key bit x (x = 63 to 32)

This write-only bitfield contains the bits [63:32] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K3, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[31:0]**: Key bit x (x = 31 to 0)

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K3, with parity bits unused.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.17 CRYP initialization vector register 0L (CRYP_IV0LR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector bit x (x = 127 to 96)

This bitfield stores the initialization vector bits [127:96] for AES chaining modes other than ECB. In DES/TDES mode it corresponds to IVI bits [63:32].

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.18 CRYP initialization vector register 0R (CRYP_IV0RR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector bit x (x = 95 to 64)

This bitfield stores the initialization vector bits [95:64] for AES chaining modes other than ECB. In DES/TDES mode it corresponds to IVI bits [31:0].

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.19 CRYP initialization vector register 1L (CRYP_IV1LR)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector bit x (x = 63 to 32)

This bitfield stores the initialization vector bits [63:32] for AES chaining modes other than ECB. *This register is not used in DES/TDES mode*.

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.20 CRYP initialization vector register 1R (CRYP_IV1RR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector bit x (x = 31 to 0)

This bitfield stores the initialization vector bits [31:0] for AES chaining modes other than ECB. *This register is not used in DES/TDES mode*.

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)

Address offset: 0x50 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCMCCM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSGCMCCM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCMCCM[31:0]**: Context swap for GCM/GMAC and CCM modes

CRYP_CSGCMCCMxR registers contain the complete internal register states of the CRYP when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 49.4.8: CRYP suspend and resume operations* for more details.

CRYP_CSGCMCCMxR registers are not used in DES/TDES or other AES chaining modes than GCM, GMAC or CCM.

49.8.22 CRYP context swap GCM registers (CRYP_CSGCMxR)

Address offset: 0x70 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSGCM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCM[31:0]**: Context swap for GCM/GMAC modes

CRYP_CSGCMxR registers contain the complete internal register states of the CRYP when the GCM or GMAC processing of the current task is suspended to process a higher-priority task. Refer to *Section 49.4.8: CRYP suspend and resume operations* for more details.

CRYP_CSGCMxR registers are not used in DES/TDES or other AES chaining modes than GCM or GMAC.

49.8.23 CRYP hardware configuration register (CRYP_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0231

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **CFG4[3:0]**: Hardware Generic 4
Reserved, must be kept at reset value.

Bits 11:8 **CFG3[3:0]**: Hardware Generic 3
This field returns the `dma_version` generic value (0x2).

Bits 7:4 **CFG2[3:0]**: Hardware Generic 2
This field returns the `aes_impl` generic value (0x3).

Bits 3:0 **CFG1[3:0]**: Hardware Generic 1
This field returns the `tdes_impl` generic value (0x1).

49.8.24 CRYP version register (CRYP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0026

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[3:0]				MINVER[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJVER[3:0]**: CRYP processor major version
This field returns the CRYP processor major version.

Bits 3:0 **MINVER[3:0]**: CRYP processor minor version
This field returns the CRYP processor minor version.

49.8.25 CRYP identification (CRYP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code

This field returns the identification code of the CRYP processor.

49.8.26 CRYP size identification register (CRYP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code

This field returns the size identification code of the CRYP processor as defined below:
SID[31:8] = 0xA3C5DD (fixed code)
SID[7:0] = 0x01 (1-Kbyte address decoding)

49.8.27 CRYP register map

Table 469. CRYP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRYP_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				ALGOMODE[3]	Res.	GCM_CCMPH[1:0]		CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
	Reset value	0						0	0	0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	KERF	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																									0	0		0	0	0	1	1

Table 469. CRYP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	CRYP_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
	Reset value																															0	0
0x20	CRYP_K0LR	K[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	K[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
...																																	
0x38	CRYP_K3LR	K[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	K[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 + x * 0x4, (x = 0 to 7), last address 0x6C	CRYP_ CSGCMCCMxR	CSGCMCCMx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 469. CRYP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x70 + x* 0x4, (x = 0 to 7), last address 0x8C	CRYP_ CSGCMxR	CSGCMx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90 to 0x3EC	Reserved	Reserved																															
0x03F0	CRYP_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value																					0	0	0	1	0	0	1	1	0	0	0	1
0x03F4	CRYP_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER [3:0]				MINVER [3:0]			
	Reset value																									0	0	1	0	0	0	1	1
0x03F8	CRYP_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x03FC	CRYP_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

50 Hash processor (HASH)

50.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family, SHA-3 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256, 384 and 512 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

50.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
  - Federal Information Processing Standards Publication FIPS PUB 202, *Secure Hash Standard* (SHA-3 family)
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for the six Keccak-based functions defined in FIPS 202 standard
  - Fixed output length: SHA3-224, SHA3-256, SHA3-384 and SHA3-512
  - Extendable-output functions: SHAKE128 and SHAKE256
  - One cycle per Keccak round, hence 58 cycles per 1088-bit block for SHA3-256
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

- 50 × 32-bit words (HR0 to HR41) for output message digest and general purpose SHA-3 outputs
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

50.3 HASH implementation

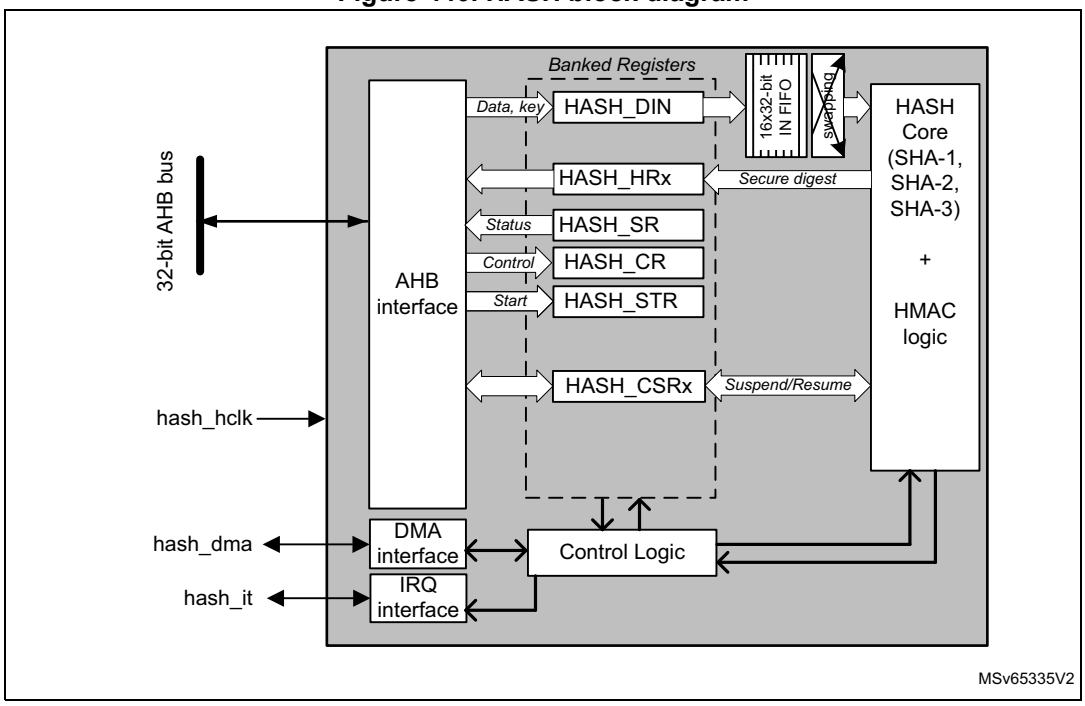
The devices have two instances of HASH peripheral.

50.4 HASH functional description

50.4.1 HASH block diagram

Figure 440 shows the block diagram of the hash processor.

Figure 440. HASH block diagram



50.4.2 HASH internal signals

Table 470 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 470. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

50.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 and FIPS PUB 202 standards.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 471*).

Table 471. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			
SHA3-224	224	144	Any length	No(3)
SHA3-256	256	136		
SHA3-384	384	104		
SHA3-512	512	72		
SHAKE128	Variable output(4)	168		
SHAKE256		136		

1. Block size = (NBWE-1) * 4 bytes. NBWE[5:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).
3. Data is fed as byte strings, not bit strings.
4. When the expected digest length is not reached (unbounded output), a new computation is required by setting DCAL in HASH_STR register. The number of computations equals the expected digest output length divided by the block size. The block size (or input rate) represents the maximum digest output that can be read after each computation.

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 and SHA-3 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

An extendable output function (XOF) SHAKE generates as many bits from its sponge as requested. For example, as per the NIST standard, for a message M and an output length d:

- SHAKE128(M,d = 256) can be used as a hash function delivering a 256-bit bit stream with 128-bit security strength. Arbitrarily large lengths can be used as pseudorandom number generators.
- SHAKE256(M, d = 128) can be used as a hash function with a 128-bit length and 128-bit resistance.

In compliance with FIPS PUB 202, SHAKE sponge functions provide different levels of preimage resistance (128 bits for SHAKE128, 256 bits for SHAKE256), provided the output size is no less than 128 bits for SHAKE128, and 256 bits for SHAKE256.

When the application selects a different output size, the security level is defined as follows (capacity_size = 256 for SHAKE128, 512 for SHAKE256):

- Security target against collisions = min (capacity_size / 2, output_size / 2)
- Security target against preimage attacks = min (capacity_size / 2, output_size)

50.4.4 Message data feeding

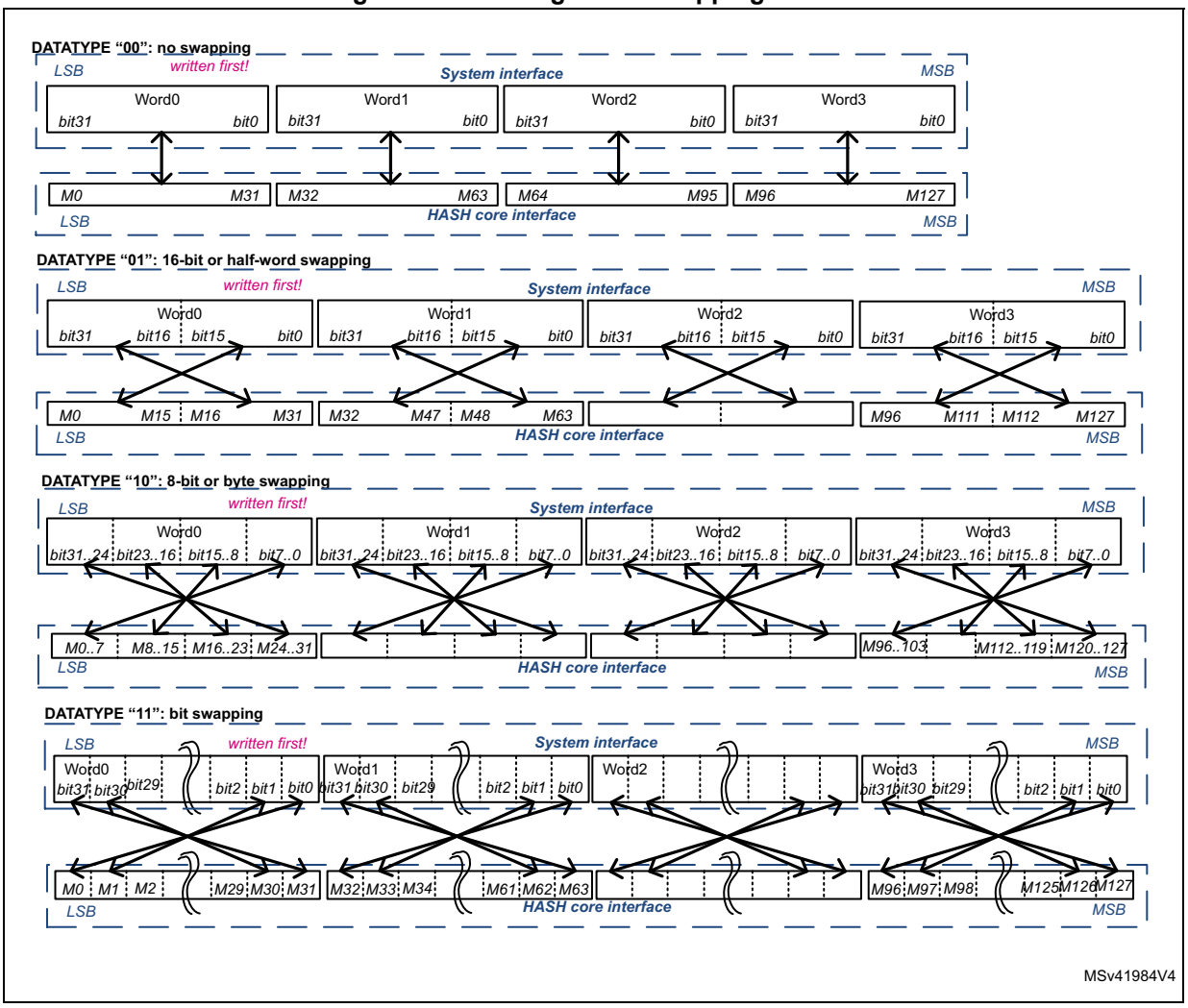
The message (or data file) to be processed by the HASH must be considered as a bit string, except for SHA-3 modes (see *Table 471*) where the message is a byte string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 441* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 441. Message data swapping feature



50.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 471: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 50.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   – Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **(NBWE-1) * 4**. This step is not required if the block size is already known (see *Table 471* for details).
   – Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   – When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 50.4.7: HMAC operation* for details.
   – Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   – Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 50.4.6* for details).
   – Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 50.4.6* for details).
   – Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA

transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 472*. When set, the DMAEN bit is automatically cleared by hardware.
5. For SHAKE128 or SHAKE256, a single computation may not be not sufficient (actual number of computation = expected output length divided by the block size). For each additional computation, execute the sequence below in CPU mode after step 4:
   a) Set DCAL in HASH_STR register.
   b) Wait for DCIS = 1, then read the new resulting digest from the same output registers. If an additional computation is required, go to step a.

Table 472. Hash processor outputs

Algorithm	Valid output registers(1)	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(2)
SHA3-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA3-256	HASH_H0 to HASH_H7		256
SHA3-384	HASH_H0 to HASH_H11		384
SHA3-512	HASH_H0 to HASH_H15		512
SHAKE128	HASH_H0 to HASH_H41		Variable output(3)
SHAKE256	HASH_H0 to HASH_H33		

1. Aliased HASH_HRAx registers are not used with SHA-3 or SHAKE algorithm (registers read as zero).
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)
3. Unbounded in the upper limit (extendable output functions), see *Table 471*.

For more information about HMAC detailed instructions, refer to *Section 50.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

50.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Note: When a SHA-3 mode is selected in HASH_CR register, the message is defined as a number of bytes with the last word having a valid number of bytes between 1 and 4. Consequently NBLW bits 0 to 2 are read as zeros in HASH_STR register and writes to 1 are ignored.

Padding processing

Detailed hashing sequences with padding are described in *Section 50.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where '**U**' means don't care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

50.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms, except SHAKE.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into

HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.

3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 50.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 50.4.11: HASH processing time.

Endianness management details can be found in Section 50.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

```
00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
30313233 34353637 38393A3B 3C3D3E3F
```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

```
53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
6E3D626C 6F636B6C 656EUUUU
```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

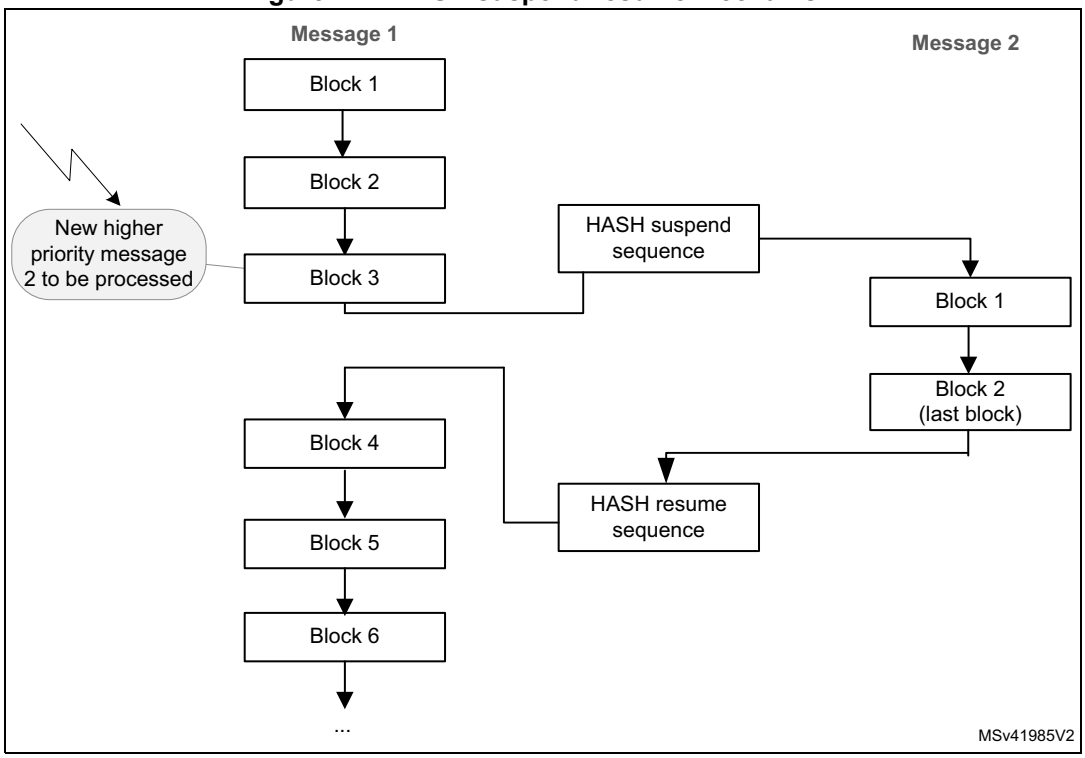
```
HASH_HR0 = 0x5FD596EE
HASH_HR1 = 0x78D5553C
HASH_HR2 = 0x8FF4E72D
HASH_HR3 = 0x266DFD19
HASH_HR4 = 0x2366DA29
```

50.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 442*.

Figure 442. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.
   - HASH_CSR0 to HASH_CSR71, when SHA-3 is selected, plus HASH_CSR72 to HASH_CSR87 if a SHA3-384 or SHA3-512 HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 442*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in

HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.

5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.
   - When a SHA-3 algorithm is selected, save HASH_CSR0 to HASH_CSR71, plus HASH_CSR72 to HASH_CSR87 if a SHA3-384 or SHA3-512 HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

50.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 50.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

Note: While a SHA-3 algorithm is selected in HASH_CR register bits 0 to 2 of NBLW are always zero (processing of bytes, not bits).

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

50.4.10 HASH error management

No error flags are generated by the hash processor.

50.4.11 HASH processing time

Table 473 summarizes the time required to process an intermediate block for each mode of operation.

Table 473. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				
SHA3-224	144	36	24(3)	**60**
SHA3-256	136	34		**58**
SHA3-384	104	26		**50**
SHA3-512	72	18		**42**

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.
3. One cycle per round, 24 rounds required for these operations.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

50.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can

individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 474* gives a summary of the available features.

Table 474. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

50.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

50.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11 when ALGO = 0x00XX.

0000: SHA-1
0001: reserved
0010: SHA2-224
0011: SHA2-256
1100: SHA2-384
1101: SHA2-512/224
1110: SHA2-512/256
1111: SHA2-512
0100: SHA3-224
0101: SHA3-256
0110: SHA3-384
0111: SHA3-512
1000: SHAKE128
1001: SHAKE256
Others: reserved, must not be used

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 471: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.

10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.

11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.

Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.

Setting INIT bit does not clear DMAE bit.

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

Bits 1:0 Reserved, must be kept at reset value.

50.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 471: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.
Do not write to this register when BUSY is set.
Reading this register returns zeros.

50.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note: When the SHA-3 family is selected, bits 0 to 2 are read as zeros and writes to 1 are ignored (byte oriented input).

50.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 or SHA3-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 or SHA3-256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384, or SHA3-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512, or SHA3-512 digest result.
- HASH_HR0 to HASH_HR41 registers return the SHAKE128, while HASH_HR0 to HASH_HR33 registers return the SHAKE256 digest result (see for *Table 472: Hash processor outputs* for details).

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing), except when the SHA-3 or SHAKE algorithm is selected (aliased registers read as zero).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 41)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

50.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
- 0: Digest calculation completion interrupt disabled
- 1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
- 0: Data input interrupt disabled
- 1: Data input interrupt enabled

50.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	NBWP[5:0]						Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r	r	r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **NBWE[5:0]:** Number of words expected

This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.

NBWE is set to the expected block size +1 in words when INIT bit is set in HASH_CR. It is set to the expected block size when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

Bits 14:9 **NBWP[5:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed

1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing

1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer

1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

50.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]:** Context swap x
Refer to *Section 50.6.7: HASH context swap registers* introduction.

50.6.8 HASH hardware configuration register (HASH_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0001 0113

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG5[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **CFG5[3:0]**: HW generic 5
This bitfield reflects that the SHA2-384 and SHA2-512 algorithms are supported.

Bits 15:12 **CFG4[3:0]**: HW generic 4
This bitfield is read as zero.

Bits 11:8 **CFG3[3:0]**: HW generic 3
This bitfield reflects that the SHA-3 algorithm is supported.

Bits 7:4 **CFG2[3:0]**: HW generic 2
This bitfield reflects that the HMAC is supported.

Bits 3:0 **CFG1[3:0]**: HW generic 1
this bitfield reflects the DMA type connected to this HASH peripheral.

50.6.9 HASH version register (HASH_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits return the HASH major revision

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits return the HASH minor revision

50.6.10 HASH identification register (HASH_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier
These bits return the unique identifier of the HASH peripheral.

50.6.11 HASH size identification register (HASH_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code

This bitfield returns the size identification code of the HASH peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1-Kbyte address decoding)

50.6.12 HASH register map

Table 475. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[5]	NBWE[4:0]					DINNE	NBWP[5]	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value											0	1	0	0	0	1	0	0	0	0	0	0	0						0	0	0	1
Reserved																																	

Table 475. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 41) Last address: 0x3B4	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x3F0	HASH_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value	0x0001 0113																															
0x3F4	HASH_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	0	1
0x3F8	HASH_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
0x3FC	HASH_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

51 On-the-fly decryption engine (OTFDEC)

51.1 OTFDEC introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the flash memory using the memory-mapped mode of the flash memory controller.

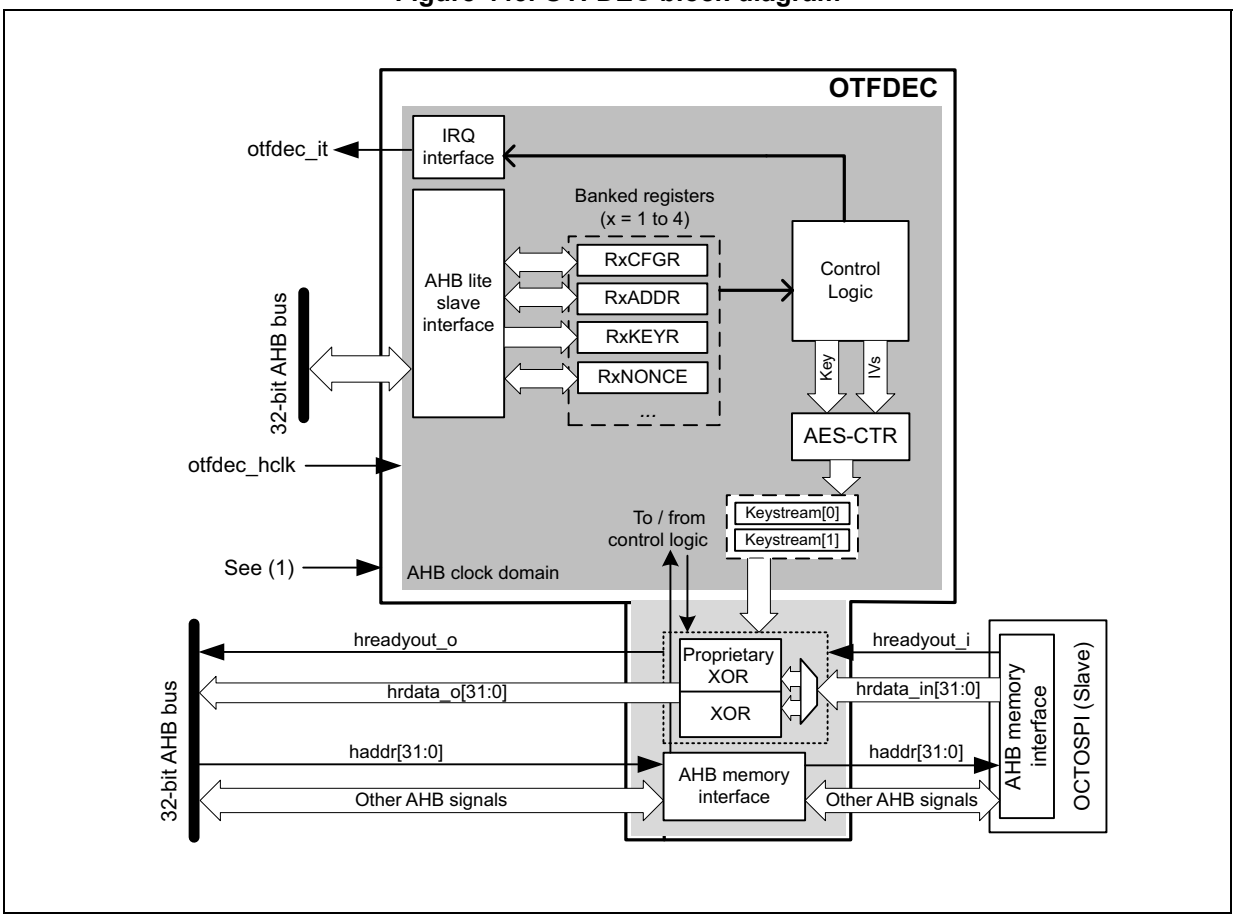
51.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Encryption mode

51.3 OTFDEC functional description

51.3.1 OTFDEC block diagram

Figure 443. OTFDEC block diagram



1. Empty (no signal)

51.3.2 OTFDEC internal signals

Table 476 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 476. OTFDEC internal input/output signals

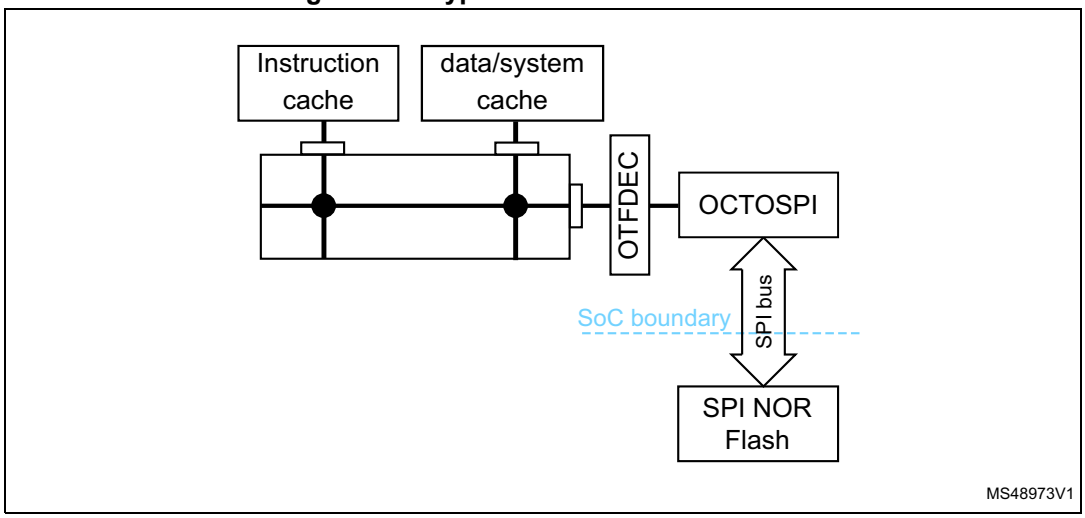
Signal name	Signal type	Description
otfdec_hclk	Digital input	AHB bus clock
otfdec_it	Digital output	OTFDEC global interrupt request

51.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 444*.

Figure 444. Typical OTFDEC use in a SoC



Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the flash memory using the Memory-mapped mode of the flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see *Section 51.5.3: Encrypting for OTFDEC* for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

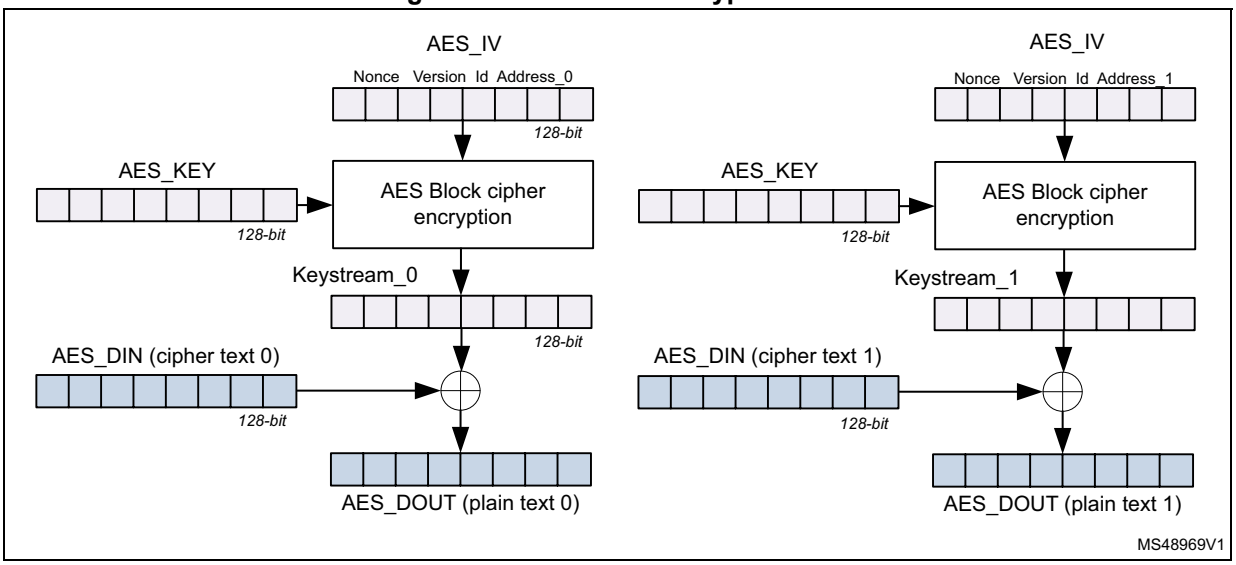
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

51.3.4 OTFDEC usage of AES in counter mode decryption

Figure 445 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Figure 445. AES CTR decryption flow



Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector AES_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4]
- key material AES_KEY[127:0] = R*x*KEYR3[31:0] || RxKEYR2[31:0] || R*x*KEYR1[31:0] || R*x*KEYR0[31:0]

Note: Above x is the RegionID of the selected encrypted region (x=1 to 4).
ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- AES_DIN and AES_DOUT data blocks are constructed following the rule below (“|” represents a binary concatenation):
  AES_Dx[127:0]= AHB_word(@ | 0xC)[31:0] | AHB_word(@ | 0x8)[31:0] | AHB_word(@ | 0x4)[31:0] || AHB_word(@ | 0x0)[31:0], where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).

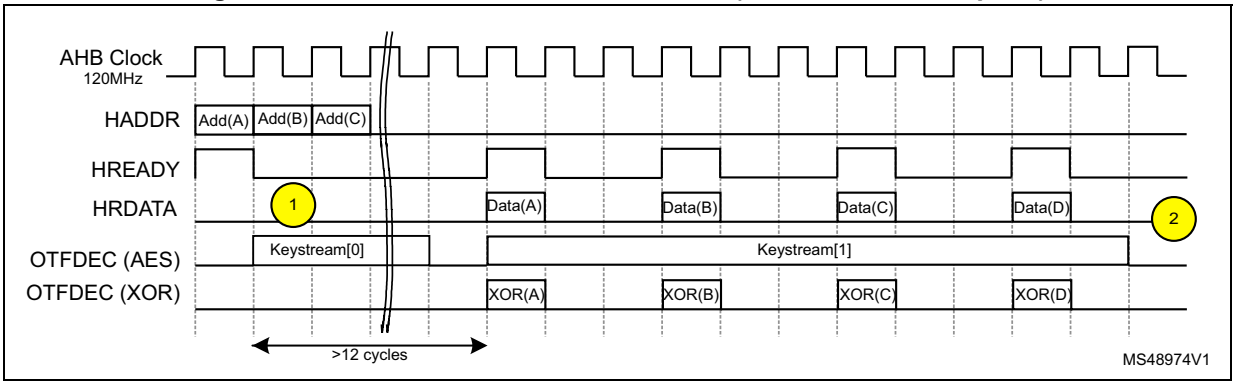
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

51.3.5 Flow control management

Figure 446 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 446. OTFDEC flow control overview (dual burst read request)



with the following notes:

1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

51.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4).
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC = 0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event (tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

For these errors, an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see *Section 51.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

51.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see *Section 51.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 51.3.6*
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see *Section 51.3.6*.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in *Table 477*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 477. OTFDEC interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
OTFDEC	Security error	SEIF	SEIE	Set SEIF in OTFDEC_ICR
	Execute-only Execute while encryption	XONEIF	XONEIE	Set XONEIF in OTFDEC_ICR
	Key error	KEIF	KEIE	Set KEIF in OTFDEC_ICR

1. The event flags are found in the OTFDEC_ISR register.

51.5 OTFDEC application information

51.5.1 OTFDEC initialization process

Introduction

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.

Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration.

Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For *x* = 1 to 4, write the correct MODE[1:0] value in OTFDEC_RxCFGR.
2. For *x* = 1 to 4, program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For *x* = 1 to 4, check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers in this region *x* are no more writable).
4. To do to decrypt a region x (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and MODE bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region x (this task **must** be performed by the entity that owns the corresponding key):
   a) Write the correct MODE[1:0] value in OTFDEC_R*x*CFGR.
   b) Program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG_EN.
   f) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

51.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 51.5.1* must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

51.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in *Section 51.3.4: OTFDEC usage of AES in counter mode decryption*.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR.
2. Encryption application initializes OTFDEC as described in *Section 51.5.1: OTFDEC initialization process*. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.
3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.
4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.
5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external flash.

There are few important notes about this procedure:

- Encryption granularity is 32-bit (single 32-bit access is mandatory).
- While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOPSI while ENC bit is set.
- OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.

51.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```
uint8_t getCRC(uint32_t * keyin)
{
  const uint8_t CRC7_POLY = 0x7;
  const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
  uint8_t  i, j, k, crc = 0x0;
  uint32_t  keyval;

  for (j = 0; j < 4; j++)
  {
    keyval = *(keyin+j);
    if (j == 0)
    {
      keyval ^= key_strobe[0];
    }
    else
    {
      keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc;
    }

    for (i = 0, crc = 0; i < 32; i++)
    {
      k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
      crc <<= 1;
      if (k)
      {
        crc ^= CRC7_POLY;
       }
    }
    crc^=0x55;
  }
  return crc;
}
```

51.6 OTFDEC registers

51.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENC**: Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode

1: OTFDEC working in encryption mode

Note: When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).

51.6.2 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Writes are ignored if CONFIGLOCK bit is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_VERSION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYCRC[7:0]								Res.	Res.	MODE[1:0]		Res.	KEYLOCK	CONFIGLOCK	REG_EN
r	r	r	r	r	r	r	r			rw	rw		rs	rs	rw

Bits 31:16 **REG_VERSION[15:0]**: region firmware version

This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]**: region key 8-bit CRC

When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.

When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in *Section 51.5.4*.

This field is read only.

Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: operating mode

This bitfield selects the OTFDEC operating mode for this region:

10: All read accesses are decrypted (instruction or data).

11: Enhanced encryption mode is activated, and only instruction accesses are decrypted

Others: Reserved

When MODE ≠ 11, the standard AES encryption mode is activated.

When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: region key lock

0: Writes to this region KEYRx registers are allowed.

1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.

Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK**: region config lock

0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.

1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.

Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN**: region on-the-fly decryption enable

0: On-the-fly decryption is disabled for this region.

1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.

Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.

51.6.3 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_START_ADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_START_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_START_ADDR[31:0]**: Region AHB start address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros .

51.6.4 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_END_ADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_END_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_END_ADDR[31:0]**: Region AHB end address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set, and OTFDEC_RxENDADDR must be strictly greater than OTFDEC_RxSTARTADDR to be valid.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in OTFDEC_RxCFGR is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones.

51.6.5 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[31:0]**: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.

51.6.6 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

51.6.7 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.

Note: When application successfully changes MODE bits in OTFDEC*_RxCFGR and* OTFDEC*_RxKEYR, and associated KEYCRC are erased.*

51.6.8 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

51.6.9 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

51.6.10 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

51.6.11 OTFDEC interrupt status register (OTFDEC_ISR)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.

1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No execute-only error status. No interrupt pending.

1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF:** Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No security error status. No interrupt pending.

1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

51.6.12 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear

This bit is written by application, and always read as 0.

0: KEIF flag status is not affected.

1: KEIF flag status is cleared in OTFDEC_ISR.

Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear

This bit is written by application, and always read as 0.

0: XONEIF flag status is not affected.

1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear

This bit is written by application, and always read as 0.

0: SEIF flag status is not affected.

1: SEIF flag status is cleared in OTFDEC_ISR.

51.6.13 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.

0: Interrupt generation on key error flag KEIF is disabled (masked).

1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE:** Execute-only execute-never error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.

0: Interrupt generation on execute-only error XONEIF is disabled (masked).

1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE:** Security error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.

0: Interrupt generation on security error SEIF is disabled (masked).

1: Interrupt generation on security error SEIF is enabled (not masked).

51.6.14 OTFDEC hardware configuration register (OTFDEC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 1B0C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CFG4[3:0]				CFG3[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								CFG1[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CFG4[3:0]**: Hardware Generic 4
This field returns the low_clk_freq generic value (0x00).

Bits 23:16 **CFG3[7:0]**: Hardware Generic 3
This field returns the option_bits generic value (0x00).

Bits 15:8 **CFG2[7:0]**: Hardware Generic 2
This field returns the address_region_msb generic value (0x1B).

Bits 7:0 **CFG1[7:0]**: Hardware Generic 1
This field returns the address_region_lsb generic value (0x0C).

51.6.15 OTFDEC version register (OTFDEC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
This field returns the OTFDEC peripheral major version.

Bits 3:0 **MINREV[3:0]**: Minor revision
This field returns the OTFDEC peripheral minor version.

51.6.16 OTFDEC identification register (OTFDEC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0051

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code
This field returns the identification code of the OTFDEC (AHB) peripheral.

51.6.17 OTFDEC size identification register (OTFDEC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code

This field returns the size identification code of the OTFDEC peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1KB address decoding)

51.6.18 OTFDEC register map

Table 478. OTFDEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	OTFDEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
	Reset value																																0
0x04-0x1C	Reserved	Reserved																															
0x20	OTFDEC_R1CFGR1	REG1_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x24	OTFDEC_R1STARTADDR	REG1_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	OTFDEC_R1ENDADDR	REG1_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	OTFDEC_R1NONCER0	REG1_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	OTFDEC_R1NONCER1	REG1_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	OTFDEC_R1KEYR0	REG1_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	OTFDEC_R1KEYR1	REG1_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	OTFDEC_R1KEYR2	REG1_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 478. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	OTFDEC_ R1KEYR3	REG1_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44 - 0x4C	Reserved	Reserved																															
0x50	OTFDEC_ R2CFGR	REG2_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x54	OTFDEC_ R2STARTADDR	REG2_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	OTFDEC_ R2ENDADDR	REG2_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	OTFDEC_ R2NONCER0	REG2_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	OTFDEC_ R2NONCER1	REG2_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	OTFDEC_ R2KEYR0	REG2_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	OTFDEC_ R2KEYR1	REG2_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	OTFDEC_ R2KEYR2	REG2_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	OTFDEC_ R2KEYR3	REG2_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C	Reserved	Reserved																															
0x80	OTFDEC_ R3CFGR	REG3_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x84	OTFDEC_ R3STARTADDR	REG3_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	OTFDEC_ R3ENDADDR	REG3_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	OTFDEC_ R3NONCER0	REG3_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	OTFDEC_ R3NONCER1	REG3_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 478. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x94	OTFDEC_ R3KEYR0	REG3_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	OTFDEC_ R3KEYR1	REG3_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	OTFDEC_ R3KEYR2	REG3_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	OTFDEC_ R3KEYR3	REG3_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xAC	Reserved	Reserved																															
0xB0	OTFDEC_ R4CFGR	REG4_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	
0xB4	OTFDEC_ R4STARTADDR	REG4_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	OTFDEC_ R4ENDADDR	REG4_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	OTFDEC_ R4NONCER0	REG4_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0	OTFDEC_ R4NONCER1	REG4_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	OTFDEC_ R4KEYR0	REG4_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC8	OTFDEC_ R4KEYR1	REG4_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	OTFDEC_ R4KEYR2	REG4_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	OTFDEC_ R4KEYR3	REG4_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4- 0x2FC	Reserved	Reserved																															
0x300	OTFDEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x304	OTFDEC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0

Table 478. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	OTFDEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
	Reset value																														0	0	0
0x30C-0x3EC	Reserved	Reserved																															
0x3F0	OTFDEC_HWCFGR	Res.	Res.	Res.	Res.	CFG4[3:0]				CFG3[7:0]								CFG2[7:0]								CFG1[7:0]							
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0
0x3F4	OTFDEC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	OTFDEC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
0x3FC	OTFDEC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

52 Public key accelerator (PKA)

52.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, all needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

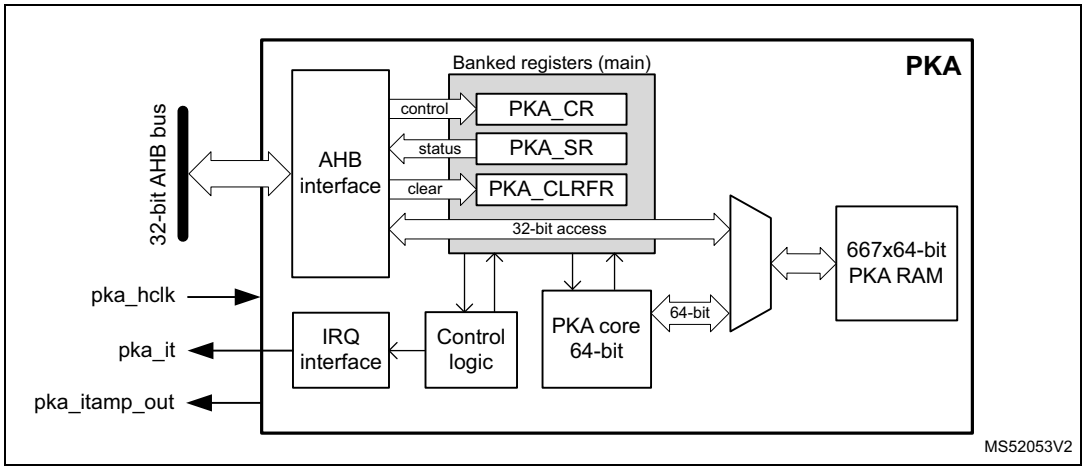
52.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

52.3 PKA functional description

52.3.1 PKA block diagram

Figure 447. PKA block diagram



52.3.2 PKA internal signals

Table 479 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 479. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

52.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 52.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

52.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 52.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 480* and *Table 481*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 480. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 52.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 52.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 52.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 52.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 52.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 52.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 52.4.7*
0x0D	001101	Modular reduction A mod n	*Section 52.4.8*
0x09	001001	Arithmetic addition A+B	*Section 52.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 52.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 52.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 52.4.12*
0x07	000111	RSA CRT exponentiation	*Section 52.4.13*

Table 481. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 52.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 52.4.15*
0x23	100011	ECC complete addition	*Section 52.4.18*
0x24	100100	ECDSA sign (protected)	*Section 52.4.16*
0x26	100110	ECDSA verification	*Section 52.4.17*
0x27	100111	ECC double base ladder	*Section 52.4.19*
0x2F	101111	ECC projective to affine	*Section 52.4.20*

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 480* and *Table 481* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in *Section 52.5.2: Computation times*.

The only operation using fast mode is modular exponentiation (MODE = 0x02).

52.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For the following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on the above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) * (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process protected version of modular exponentiation (MODE = 0x3) is strongly recommended for security reason. For encryption process MODE = 0x3 cannot be used, as it requires the knowledge of the private key.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 52.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, *n* x G = identity element O).

ECDSA message signature generation

ECDSA (elliptic curve digital signature algorithm) signature generation function principle is the following: Alice, to sign a message *m* using her private key integer d_A, goes through the following steps.

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* A^{-1} mod *n*
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A.

Bob can verify that Q_A is a valid curve point going through the following steps:

1. check that Q_A is not equal to the identity element O
2. check that Q_A is on the agreed curve
3. check that n x Q_A = O.

Then Bob follows the procedure detailed below:

1. verify that *r* and *s* are integer in [1, n-1]
2. calculate *e* = HASH(*m*), where HASH is the agreed cryptographic hash function
3. let z be the L_n leftmost bits of *e*
4. calculate w = s^{-1} mod *n*
5. calculate u_1 = zw mod *n* and u_2 = *rw* mod *n*
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if *r* = x_1 (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

52.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Note: When PKA is in the process of clearing its memory EN bit cannot be set.

When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 52.3.7).

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 52.4*.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error, and step *3* (PROCENDF = 1) never happens. See *Section 52.3.7* for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in *Section 52.3.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 52.4.5*). This is know as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 52.5.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 52.4*.

52.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the address error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM error flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return 0, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (see bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 52.6* for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

Note: The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

52.4 PKA operating modes

52.4.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.

Warning: The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified in the operation tables.

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size* / 64 + 1) words, with *rsa_size* equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand size): data size is (*ecc_size* / 64 + 1) words, with *ecc_size* equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

Note: The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation *ECC Fp scalar multiplication*, when the application writes the x coordinate of point

P for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information a = -3, on a curve with a modulus length of 224 bits (that is, four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

```
@RAM+410  0x0000000000000001  /* curve coefficient 'a' sign without extra word */
@RAM+418  0x0000000000000011  /* value of |a| LSB
@RAM+420  0x0000000000000000  ...
@RAM+428  0x0000000000000000  ...
@RAM+430  0x0000000000000000  value of |a| MSB */
@RAM+438  0x0000000000000000  /* additional all 0 word */
```

52.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod *n*) used by PKA to convert operands into the Montgomery residue system representation. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in *Table 482*.

Table 482. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	ROS
OUT	Result: R^2 mod n	-	RAM@0x620	

52.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 483*.

Table 483. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xE78	

52.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 484*.

Table 484. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xE78	

52.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Assuming that A is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – Result AR= A x *r2modn* mod *n* is A in the Montgomery domain.
   b) Assuming that BR is an integer in the Montgomery domain:
      – Result B = BR x 1 mod *n* is B in the natural domain.
      – Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n*.
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute AB= AR x B mod *n*. Output is in natural domain.
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain.
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain.
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain.
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain.
   g) (optional) Repeat the two steps above if more operands need to be multiplied.
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain.

Operation instructions for Montgomery multiplication are summarized in *Table 485*.

Table 485. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: AxB mod n(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

52.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod *n*.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 486* and *Table 487*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 488*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 486* (normal mode), *Table 487* (fast mode) and in *Table 488* (protected mode). Fast mode usage is explained in *Section 52.3.6*.

Table 486. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 487. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
IN/OUT	Montgomery parameter R2 mod n	(mandatory)	RAM@0x620	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 488. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x03	PKA_CR	6 bits
	Exponent e length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0x16C8	ROS
	Exponent e	(0 ≤ e < n)	RAM@0x14B8	
	Modulus value n	(odd integer only, n < 2^{4096})	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	
ERROR	Error A^e mod n	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n^1 with phi = (p - 1) * (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (*p*, *q*, *e*, *d*) instead of (*N*, *e*, *d*, *phi*), in this case, to derive *N* and *phi* using PKA arithmetic multiplier: *N* = *p* * *q*, and *phi* = (*p* - 1) * (*q* - 1).

52.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse A^{-1} mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the greatest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 489*.

Table 489. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xE78	

52.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 490*.

Table 490. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand A	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result A mod n	(0 < result < n)	RAM@0xE78	

52.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 491*.

Table 491. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: A+B	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

52.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \ge B$ result equals A - B
- If A < B and M/32 residue is > 0 result equals $A + 2^{int(M/32)*32+1} - B$
- If A < B and M/32 residue is 0 result equals $A + 2^{int(M/32)*32} - B$

For the last two bullets the 64-bit word following the most significant word of the output equals 0xFFFF FFFF FFFF FFFF, as result is negative.

Operation instructions are summarized in *Table 492*.

Table 492. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: A-B	($0 \le$ result $< 2^M$)	RAM@0xE78	

52.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 493*.

Table 493. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: AxB	($0 \le result < 2^{2M}$)	RAM@0xE78	2xROS

52.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in *Table 494*.

Table 494. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result A = B ?	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

52.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation $m = A^d \pmod{pq}$ more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation $A^d \bmod pq$ are summarized in *Table 495*.

Table 495. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	$(0 < d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 < d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 < q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime p(1)	$(0 < p < 2^{M/2})$	RAM@0xB60	
	Prime q(1)	$(0 < q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 < A < 2^M)$	RAM@0x12A0	ROS
OUT	Result: $A^d \bmod pq$	$(0 \le \text{result} < pq)$	RAM@0x838	

1. Must be different from 2.

52.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 496*.

Table 496. Point on elliptic curve Fp check

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < p < 2640)	RAM@0x470	
	Point P coordinate *x*	(x < *p*)	RAM@0x578	
	Point P coordinate *y*	(y < *p*)	RAM@0x5D0	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Result: point P on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: x or y coordinate is not smaller than modulus p	RAM@0x680	64 bits

52.4.15 ECC Fp scalar multiplication

This operation consists in the computation of k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 497*.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 497. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 497. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier k	($0 \leq k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x < p$)	RAM@0x578	
	Point P coordinate y_P	($y < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: k x P coordinate *x'*	(result < *p*)	RAM@0x578	
	Result: k x P coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error k x P	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For k = 0 this function returns a point at infinity (0, 0) if curve parameter b is nonzero, (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation must be carried out.
- For k < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-k| must be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

52.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 52.3.5*) is summarized in *Table 498* (input parameters) and in *Table 499* (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new k must be generated and the ECDSA sign operation must be repeated.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 498. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient b	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer k(1)	(0 ≤ k < n)	RAM@0x12A0	
	Curve base point G coordinate x	(x < *p*)	RAM@0x578	
	Curve base point G coordinate y	(y < *p*)	RAM@0x470	
	Hash of message z	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key d	(0 < d)	RAM@0xF28	
	Curve prime order n	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases k can be deterministically generated.
2. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 499. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x730	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (*Table 498* and *Table 499*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 500*.

Table 500. Extended ECDSA sign - Extra outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

52.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 52.3.5*) is summarized in *Table 501* (input parameters) and *Table 502* (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

Table 501. ECDSA verification - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0< p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 502. ECDSA verification - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– (0 < *r* < n)	RAM@0x578	EOS

52.4.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in *Table 503*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y)= (X/Z^2,Y/Z^3) can be used.

Table 503. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate X	(x < *p*)	RAM@0xD60	
	Result coordinate Y	(y < *p*)	RAM@0xDB8	
	Result coordinate Z	(z < *p*)	RAM@0xE10	

52.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized in *Table 504*.

If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 504. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer k	($0 < k < 2^{640}$)	RAM@0x520	
	Integer m	($0 < m < 2^{640}$)	RAM@0x578	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate x	(x < *p*)	RAM@0x578	EOS
	Result coordinate y	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

52.4.20 ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (X, Y, Z), it computes the affine coordinates (x, y) = (X/Z, Y/Z).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 505*.

Table 505. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate X (projective)	(x < *p*)	RAM@0xD60	
	Point P coordinate Y (projective)	(y < *p*)	RAM@0xDB8	
	Point P coordinate Z (projective)	(z < *p*)	RAM@0xE10	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Point P coordinate x (affine)	(x < *p*)	RAM@0x578	
	Point P coordinate y (affine)	(y < *p*)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

52.5 Example of configurations and processing times

52.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b \pmod{p}$.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 506 summarizes the family of curves supported by PKA for ECC operations.

Table 506. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

52.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 507. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	66000	242000	310000	503000
	Fast	7000	23000	48000	82000
17	Normal	69000	253000	333000	543000
	Fast	24000	34000	72000	123000
2^{16} + 1	Normal	82000	296000	427000	705000
	Fast	23000	78000	165000	284000
1024	Protected	2960000	-	-	-
	Normal	1725000	-	-	-
	Fast	1666000	-	-	-
	CRT(1)	595000	-	-	-
2048	Protected	-	17643000	-	-
	Normal	-	11553000	-	-
	Fast	-	11334000	-	-
	CRT(1)	-	3454000	-	-
3072	Protected	-	-	53390000	-
	Normal	-	-	36280000	-
	Fast	-	-	36019000	-
	CRT(1)	-	-	10328000	-
4096	Protected	-	-	-	119824000
	Normal	-	-	-	82619000
	Fast	-	-	-	82198000
	CRT(1)	-	-	-	23061000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield= 0x07).

Table 508. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	742000	1220000	2116000	3031000	5981000	7844000	10200000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 509. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	633000	1087000	1708000	2548000	4856000	5978000	8030000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 510. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
566000	676000	1221000	1957000	2967000	5825000	7096000	9773000

Table 511. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
541000	644000	1152000	1873000	2814000	5506000	6753000	9350000

Table 512. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
24700	32000	56000	90000	144000	271000	326000

Table 513. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
8000	8200	10500	13000	19400	29600	39000

Table 514. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
2200	2240	2870	3550	4400	6260	7460	8500

Table 515. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
4200	5200	5900	12600	35500	126500	290000	420000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

52.6 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 52.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 52.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 52.3.7*
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 516 gives a summary of the available features.

Table 516. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

52.7 PKA registers

52.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
										rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA. PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

52.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
										r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
														r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE bitfield can be used.

1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK
This bit is asserted when PKA initialization is complete.
0: PKA is not initialized correctly. START bit cannot be set.
1: PKA is initialized correctly and can be used normally.

Note: When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

52.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
										w	w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRFC**: Clear operation error flag
0: No action
1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

52.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 52.3.3: PKA reset and clocks.

52.7.5 PKA hardware configuration register (PKA_HWCFGR)

Address offset: 0x1FF0

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CFG2[3:0]**: HW Generic 2
This field returns the DPAEN generic value (0x1)

Bits 3:0 **CFG1[3:0]**: HW Generic 2
This field returns the PKA64_SEL generic value (0x1)

52.7.6 PKA version register (PKA_VERR)

Address offset: 0x1FF4

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
This field returns the PKA major version.

Bits 3:0 **MINREV[3:0]**: Minor revision
This field returns the PKA minor version.

52.7.7 PKA identification register (PKA_IPIDR)

Address offset: 0x1FF8

Reset value: 0x0017 0072

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code
This field returns the PKA identification code.

52.7.8 PKA size identification register (PKA_SIDR)

Address offset: 0x1FFC

Reset value: 0xA3C5 DD08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code
This field returns the size identification code of the PKA as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x08 (8 KB address decoding)

52.7.9 PKA register map

Table 517. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value											0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
	Reset value											0	0	0		0	0															0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0		0																	
0x00C-0x1FEC	Reserved																																
0x1FF0	PKA_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0x1FF4	PKA_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0x1FF8	PKA_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
0x1FFC	PKA_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

53 Advanced-control timers (TIM1/TIM8)

53.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 53.3.30: Timer synchronization*.

53.2 TIM1/TIM8 main features

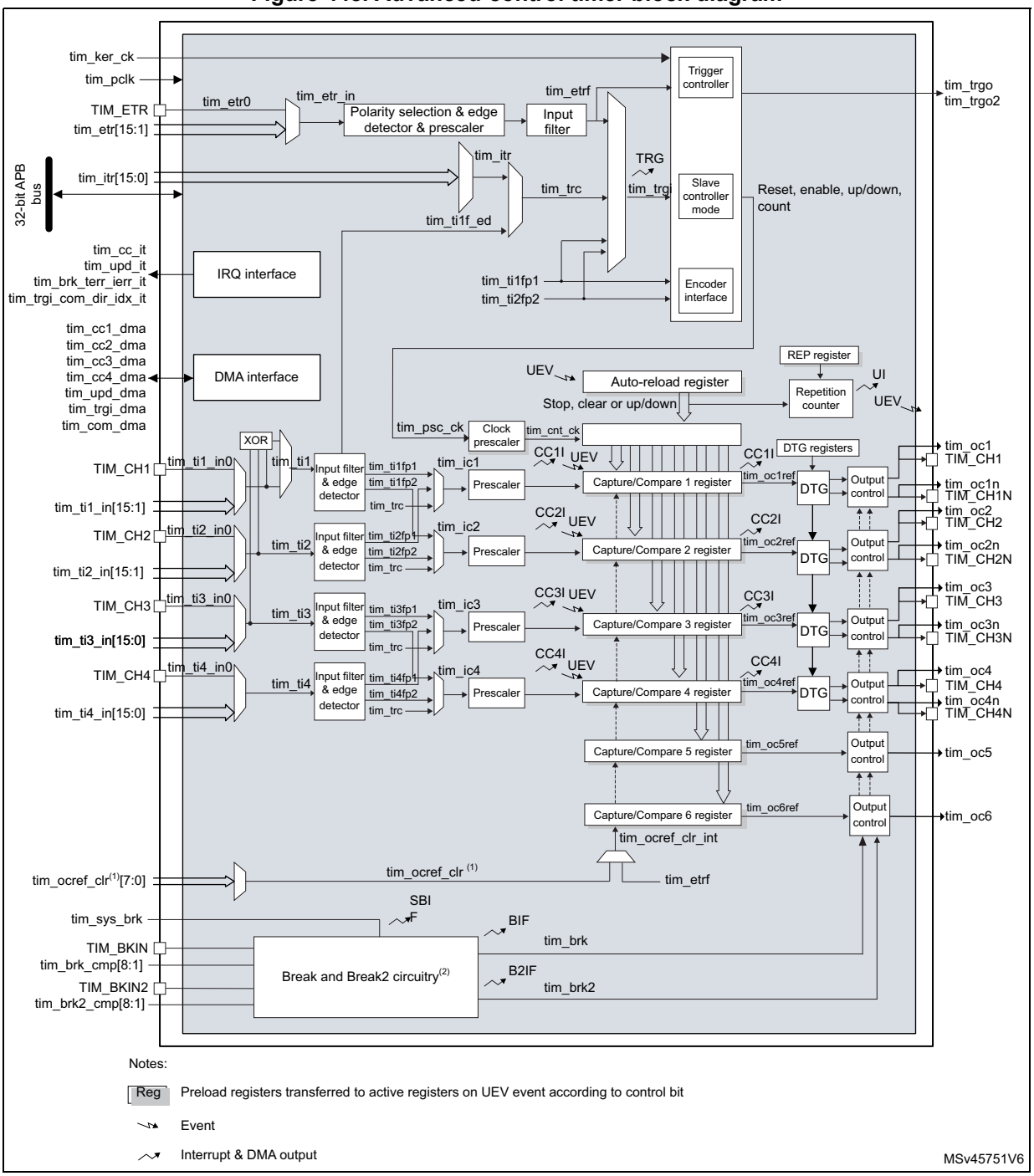
TIM1/TIM8 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

53.3 TIM1/TIM8 functional description

53.3.1 Block diagram

Figure 448. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals*.
2. See *Figure 495: Break and Break2 circuitry overview* for details.

53.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 518. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input/output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 519. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 519. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_brk2_cmp[8:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 520, *Table 521*, *Table 522* and *Table 523* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 520. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in[15:1]	Reserved	

Table 521. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[15:1]	Reserved	

Table 522. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti3_in[15:1]	Reserved	

Table 523. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in[15:1]	Reserved	

Table 524 lists the internal sources connected to the tim_itr input multiplexer.

Table 524. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	Reserved
tim_itr6	tim12_trgo	tim12_trgo
tim_itr7	tim13_oc1	tim13_oc1
tim_itr8	tim14_oc1	tim14_oc1
tim_itr9	tim15_trgo	tim15_trgo
tim_itr10	tim16_oc1	tim16_oc1
tim_itr11	tim17_oc1	tim17_oc1
tim_itr12	Reserved	
tim_itr13	tim10_oc1	tim10_oc1
tim_itr14	tim11_oc1	tim11_oc1
tim_itr15	Reserved	

Table 525 lists the internal sources connected to the tim_etr input multiplexer.

Table 525. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr1	Reserved	
tim_etr2		
tim_etr3		
tim_etr4		
tim_etr5	adc1_awd1	adc1_awd1
tim_etr6	adc1_awd2	adc1_awd2
tim_etr7	adc1_awd3	adc1_awd3
tim_etr8	adc2_awd1	adc2_awd1
tim_etr9	adc2_awd2	adc2_awd2
tim_etr10	adc2_awd3	adc2_awd3
tim_etr11	adc3_awd1	adc3_awd1
tim_etr12	adc3_awd2	adc3_awd2
tim_etr13	adc3_awd3	adc3_awd3
tim_etr[15:14]	Reserved	

Table 526, *Table 527* and *Table 528* list the sources connected to the tim_brk and tim_brk2inputs.

Table 526. Timer break interconnect

tim_brk inputs	TIM1	TIM8
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp1	Reserved	
tim_brk_cmp2		
tim_brk_cmp3		
tim_brk_cmp4		
tim_brk_cmp5		
tim_brk_cmp6		
tim_brk_cmp7		
tim_brk_cmp8	mdf1_break0	mdf1_break2

Table 527. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp1	Reserved	
tim_brk2_cmp2		
tim_brk2_cmp3		
tim_brk2_cmp4		
tim_brk2_cmp5		
tim_brk2_cmp6		
tim_brk2_cmp7		
tim_brk2_cmp8	mdf1_break1	mdf1_break3

Table 528. System break interconnect

tim_sys_brk inputs	TIM1 / TIM8	Enable bit in SYSCFG_CBR register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	Backup RAM double ECC error	BKRAML
tim_sys_brk3	Retention SRAM double ECC error	RETRAML
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

53.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 449 and *Figure 450* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 449. Counter timing diagram with prescaler division change from 1 to 2

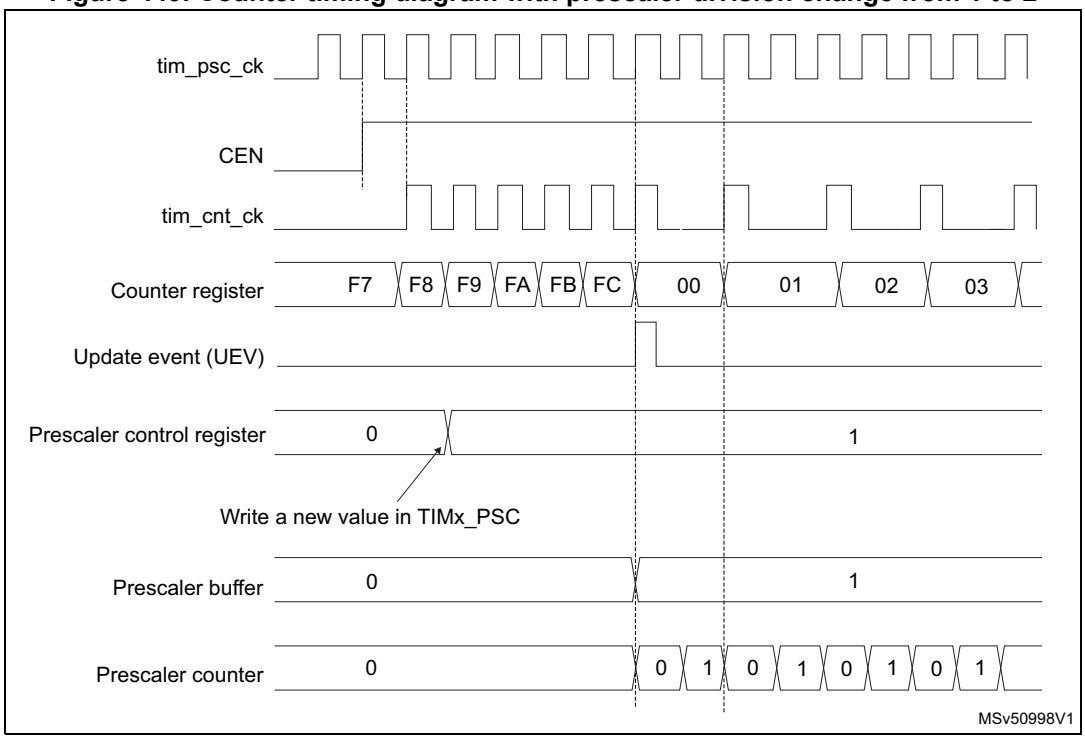
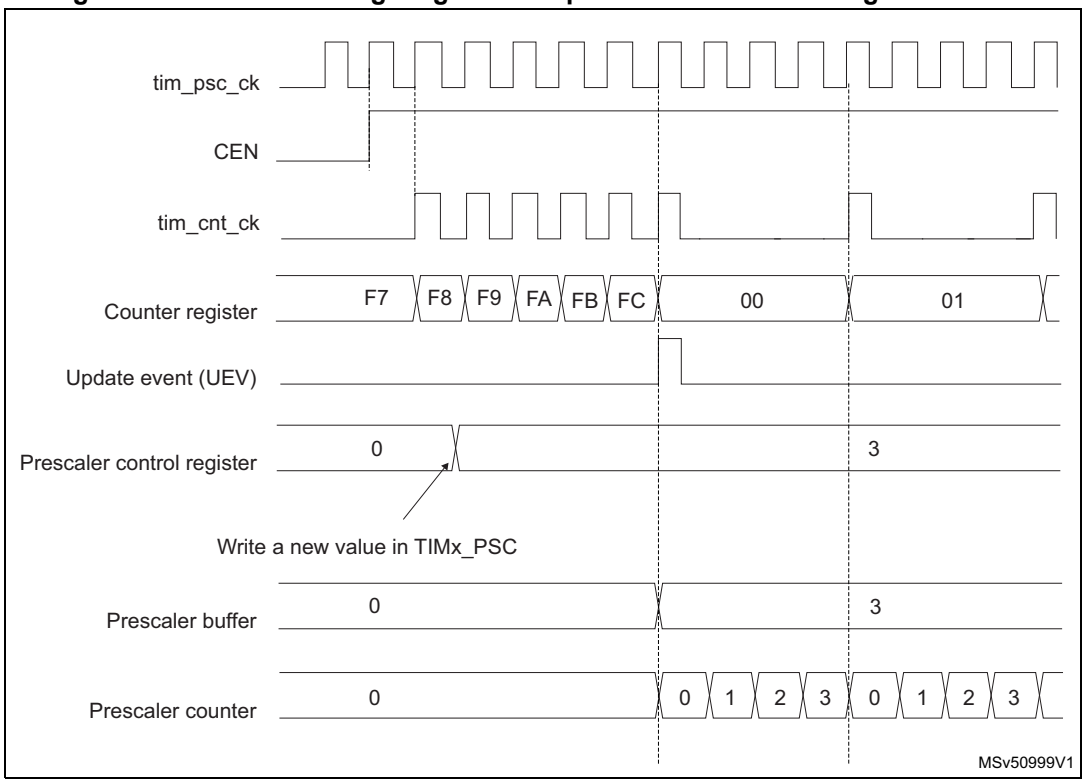


Figure 450. Counter timing diagram with prescaler division change from 1 to 4



53.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 451. Counter timing diagram, internal clock divided by 1

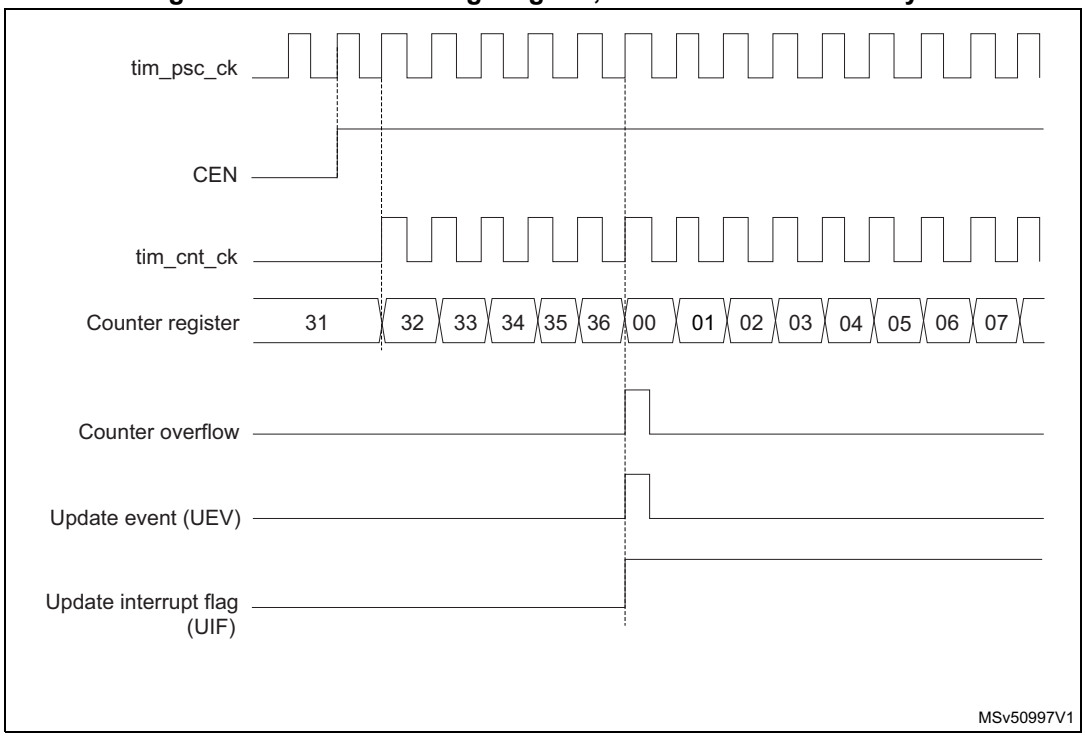


Figure 452. Counter timing diagram, internal clock divided by 2

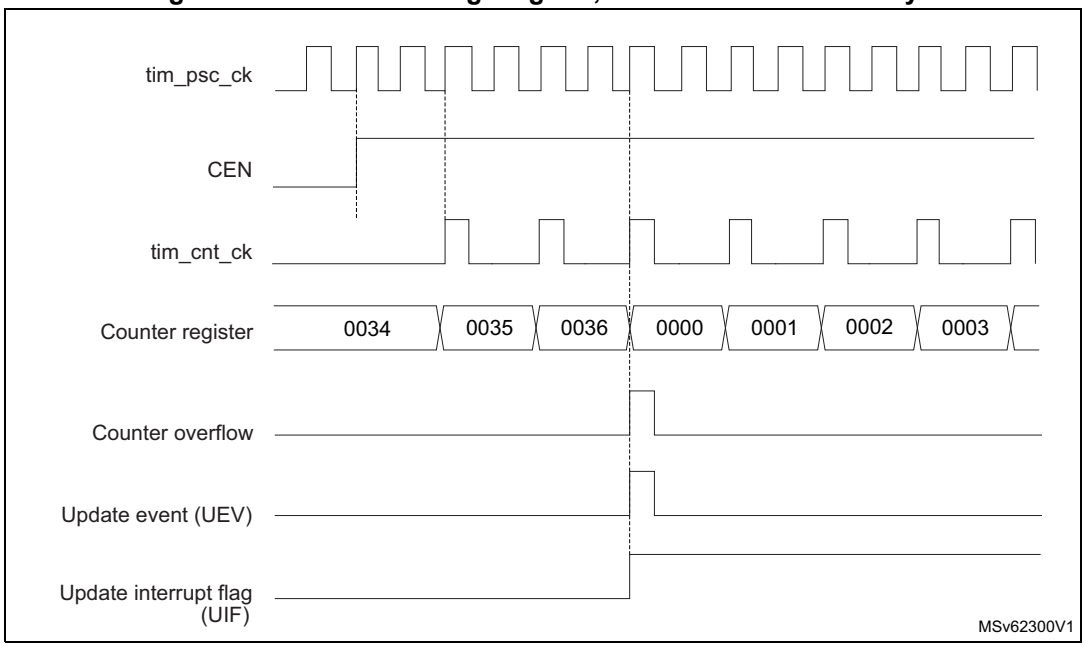


Figure 453. Counter timing diagram, internal clock divided by 4

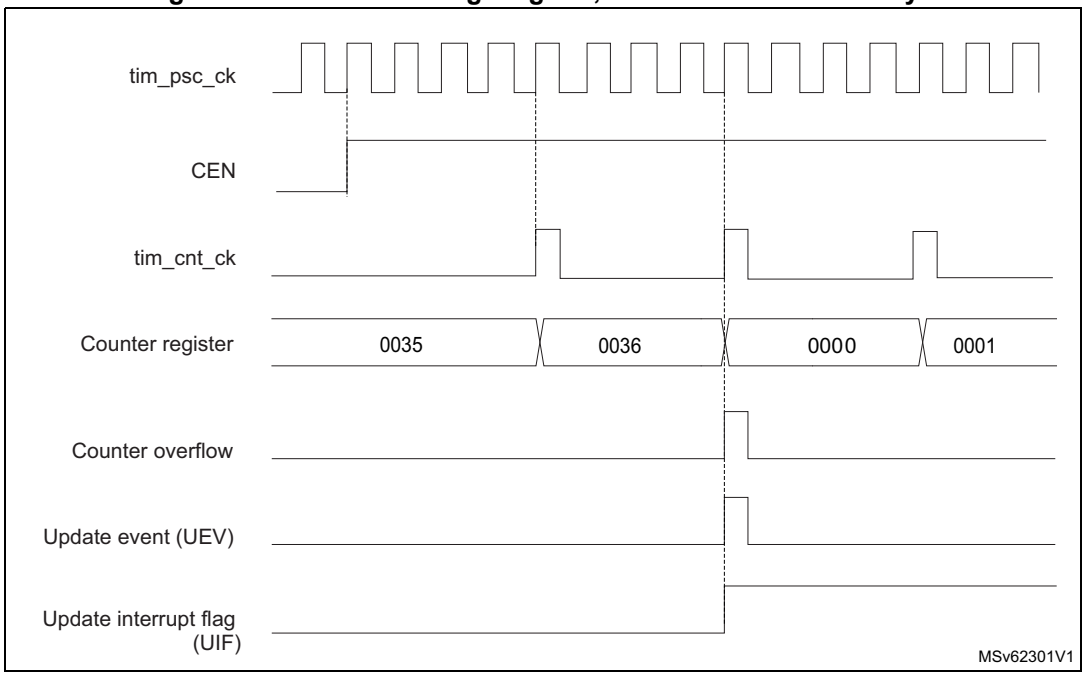


Figure 454. Counter timing diagram, internal clock divided by N

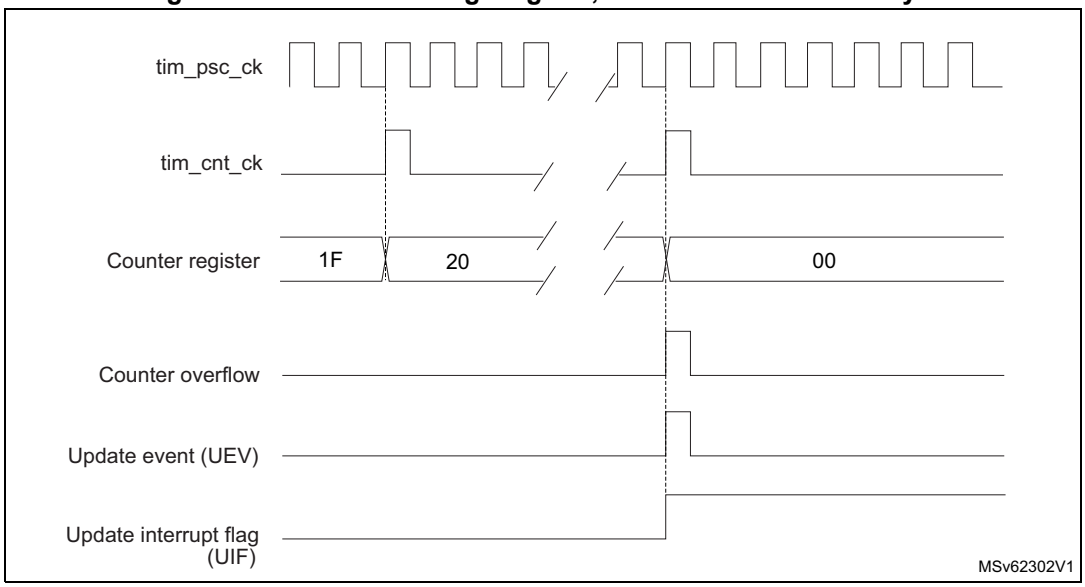


Figure 455. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

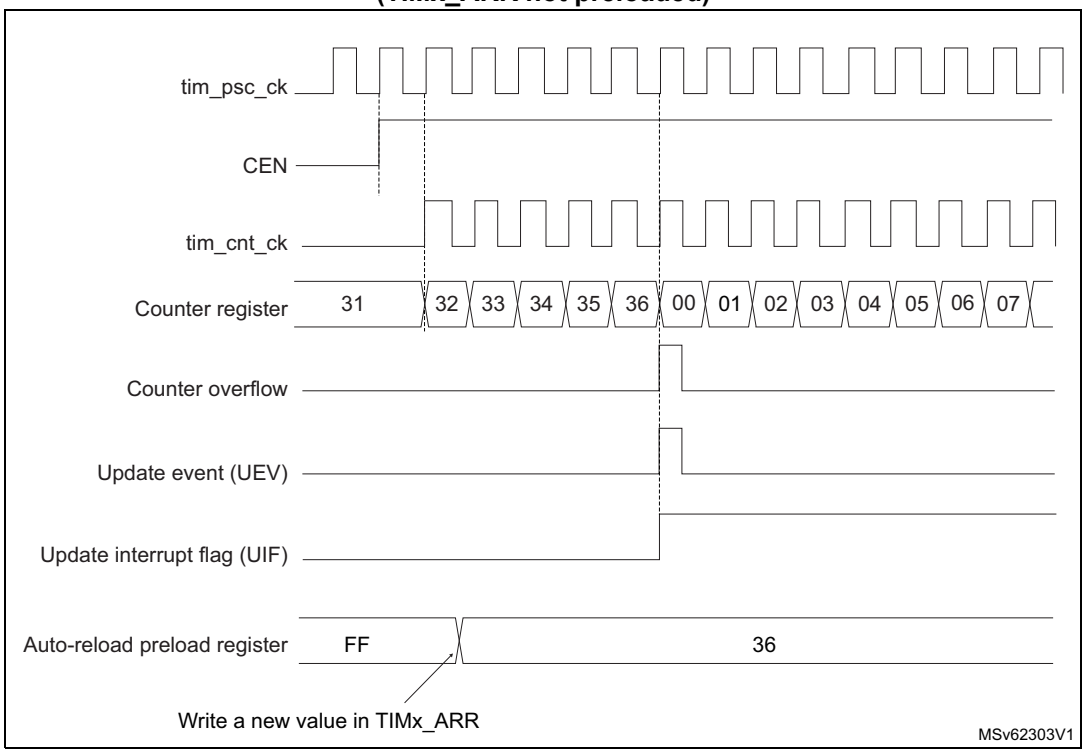
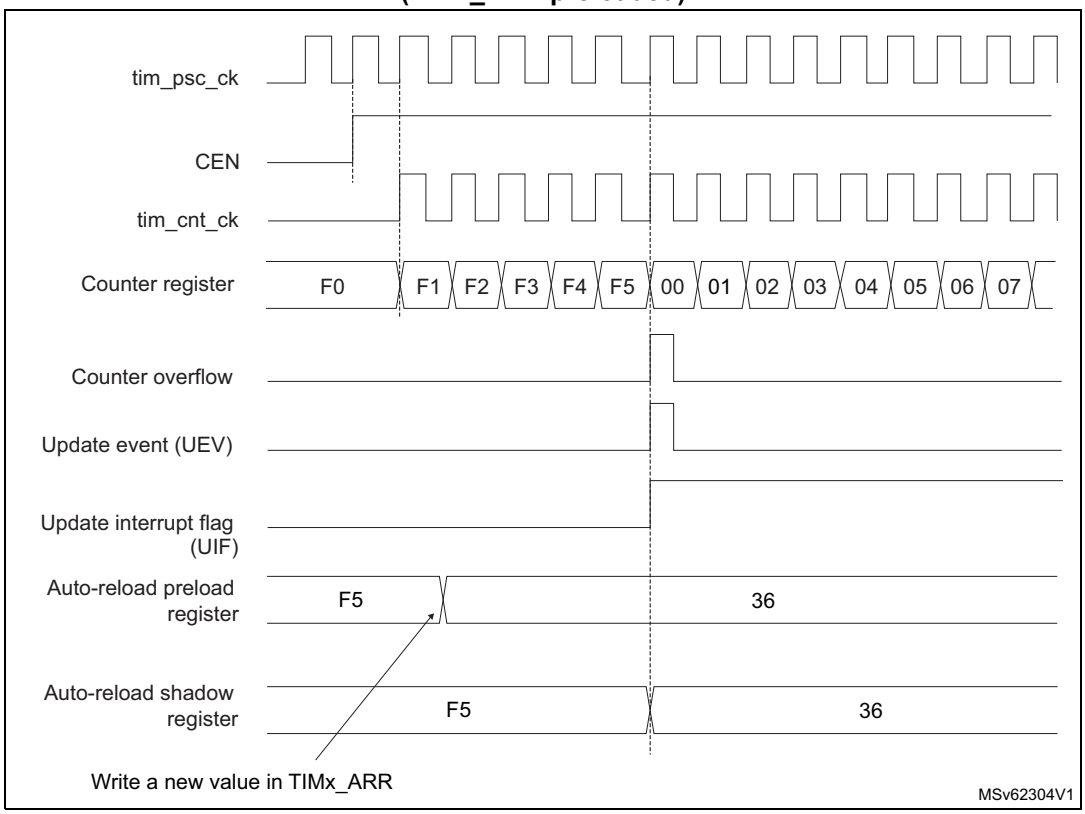


Figure 456. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 457. Counter timing diagram, internal clock divided by 1

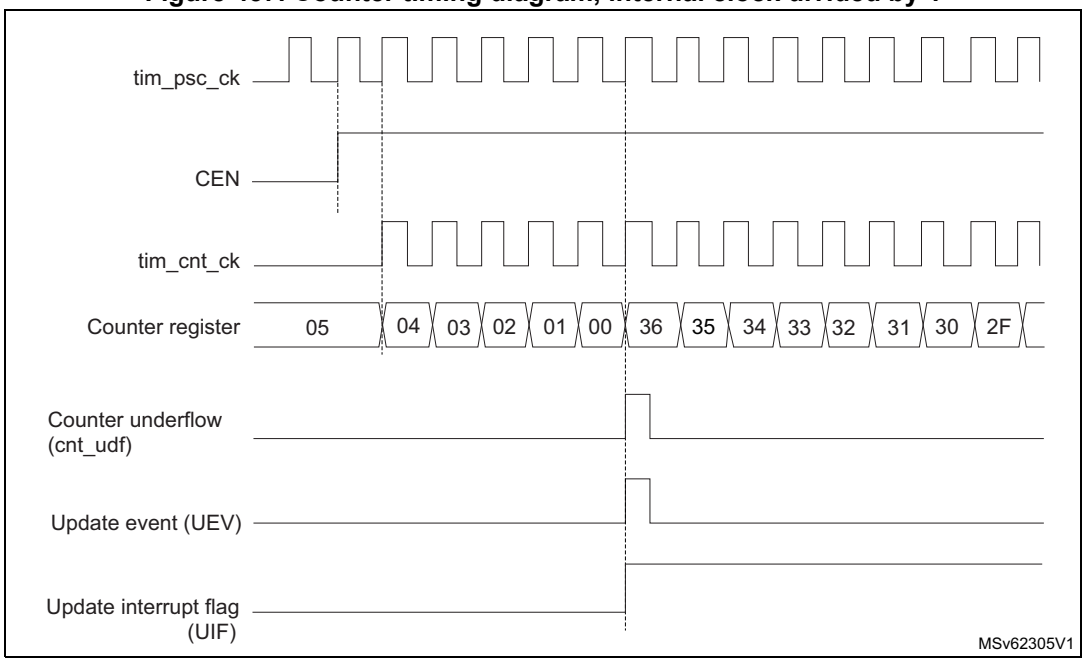


Figure 458. Counter timing diagram, internal clock divided by 2

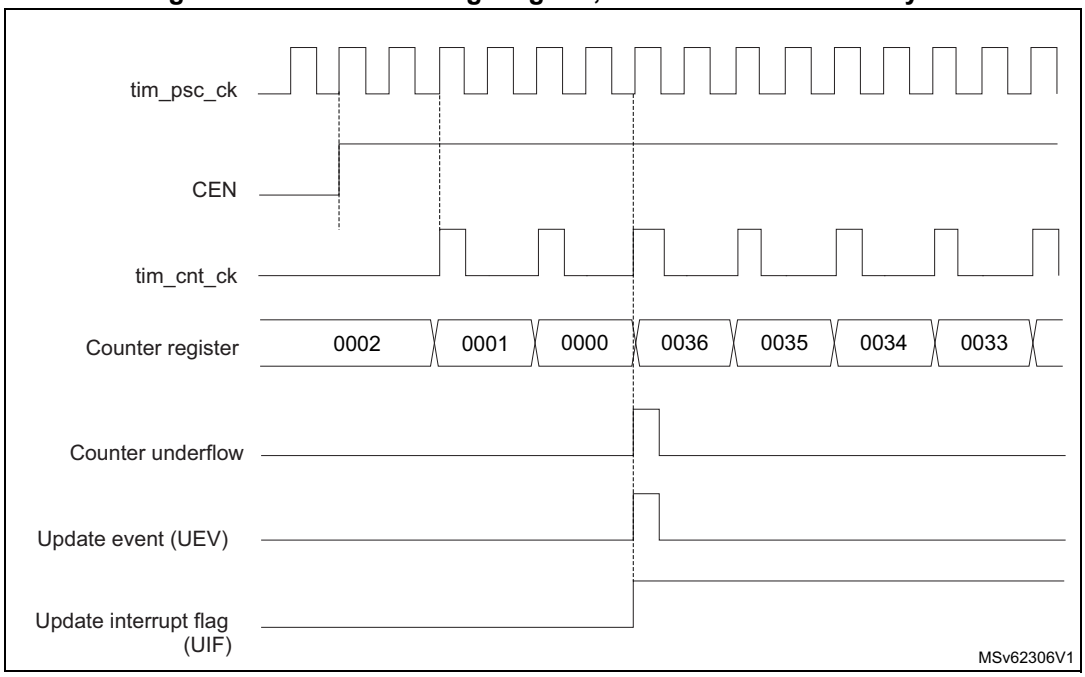


Figure 459. Counter timing diagram, internal clock divided by 4

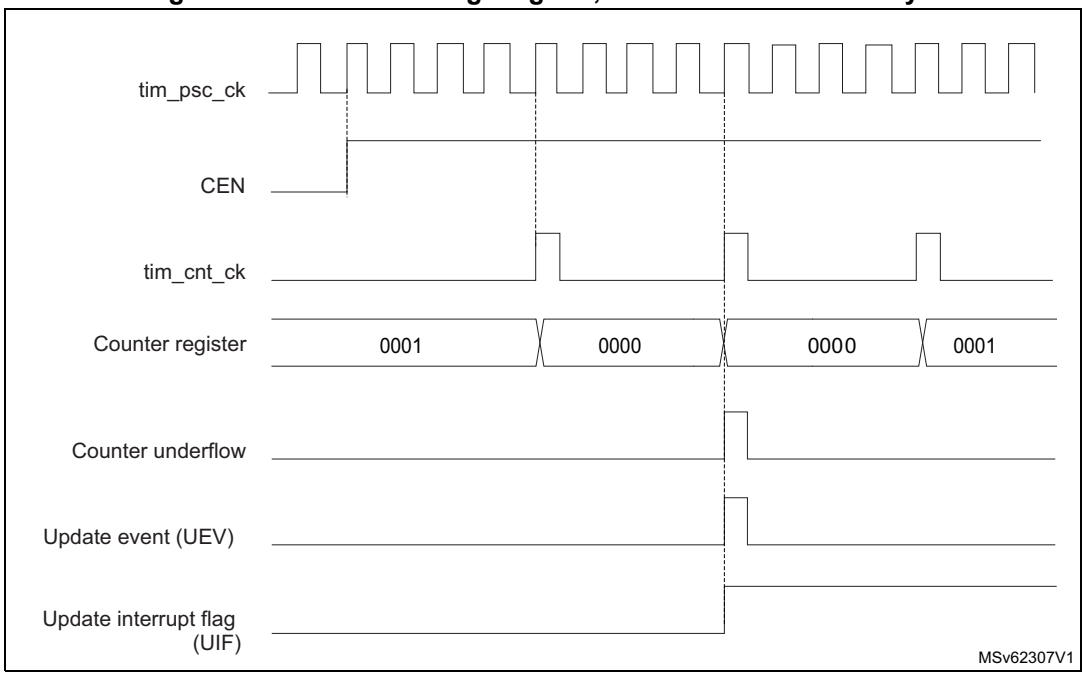


Figure 460. Counter timing diagram, internal clock divided by N

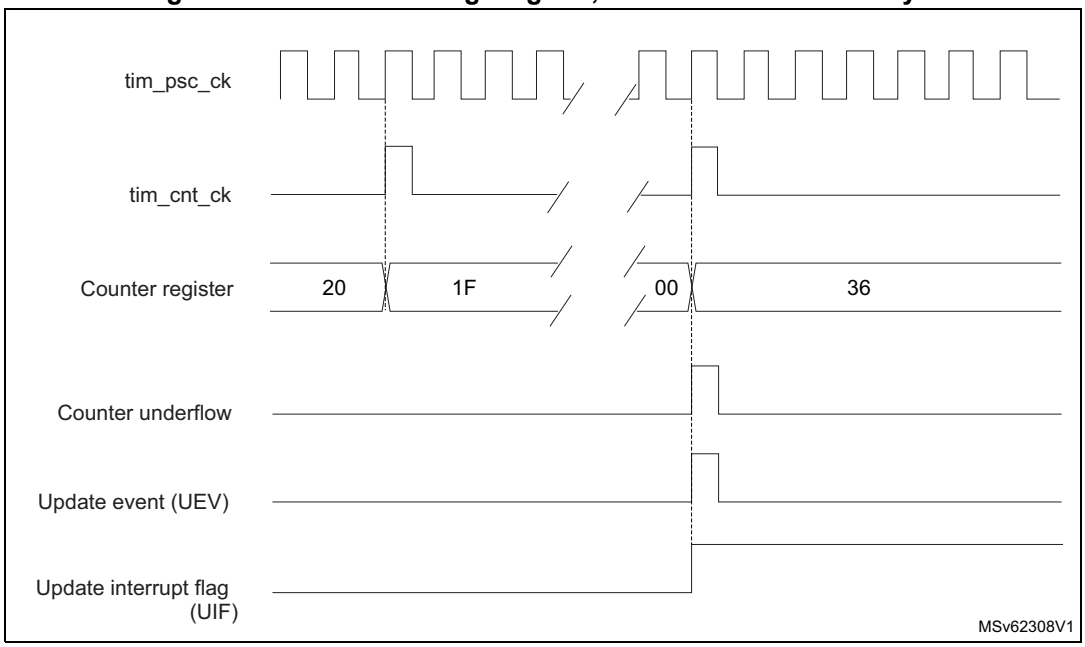
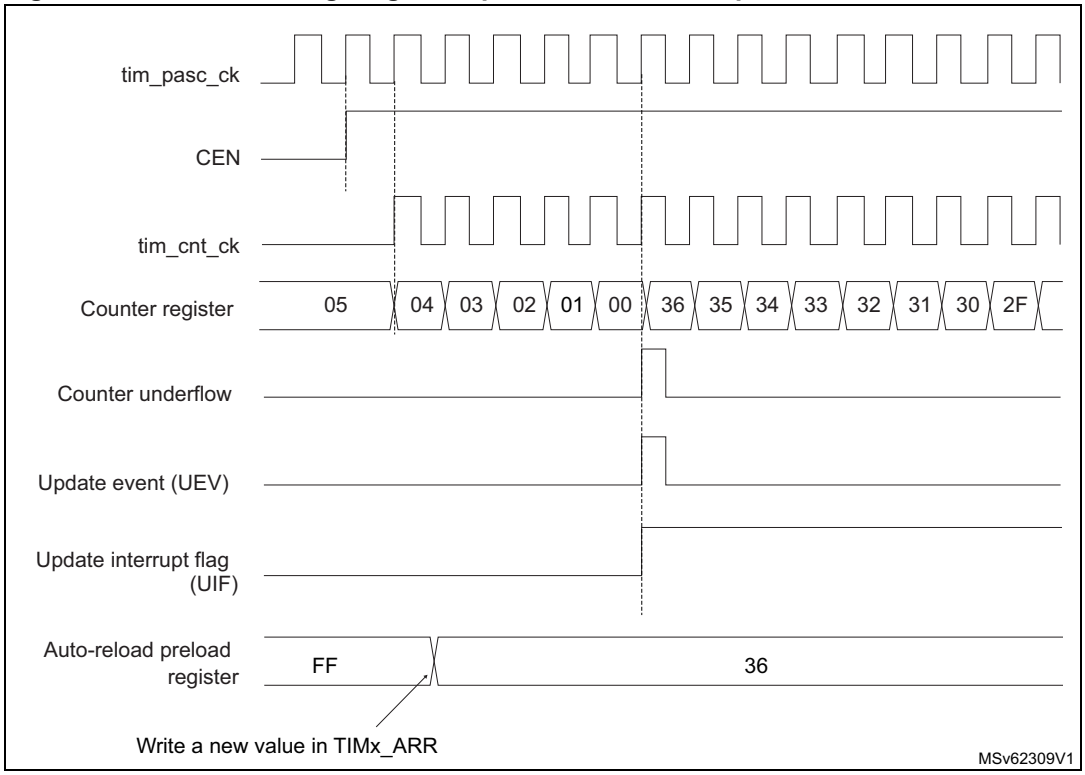


Figure 461. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

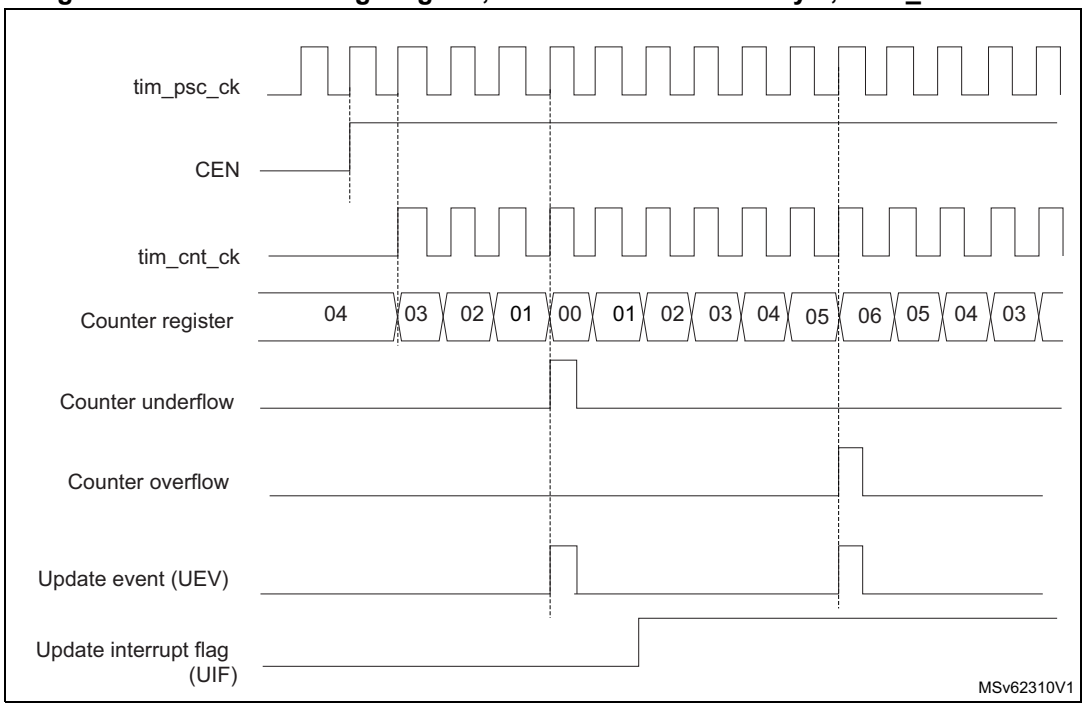
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 462. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 53.6: TIM1/TIM8 registers*).

Figure 463. Counter timing diagram, internal clock divided by 2

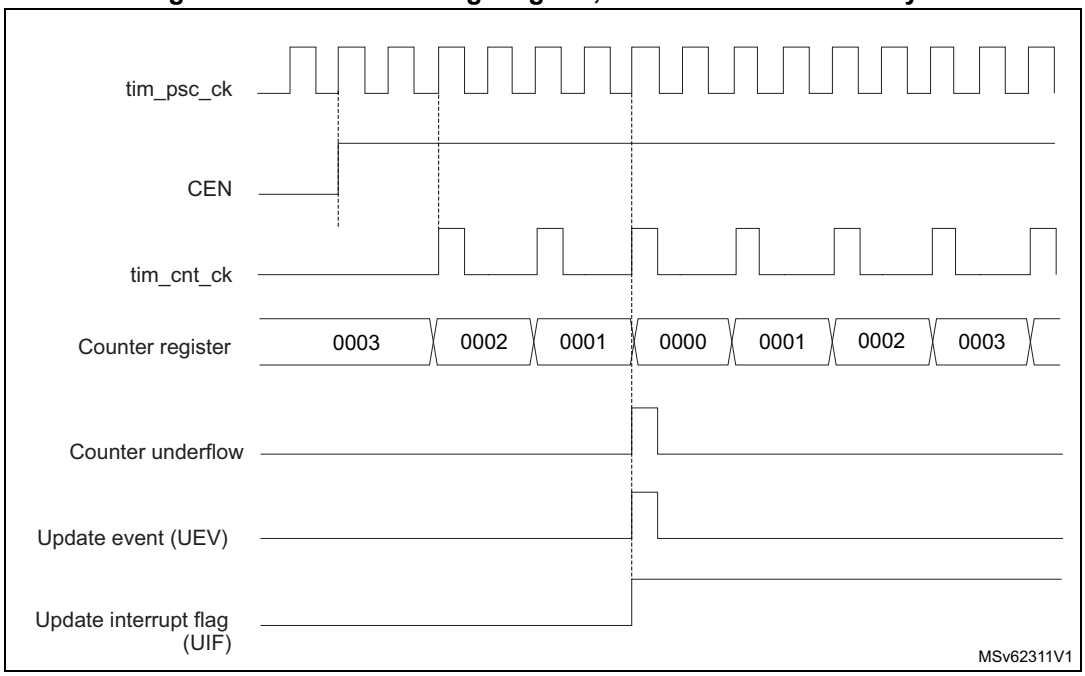


Figure 464. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

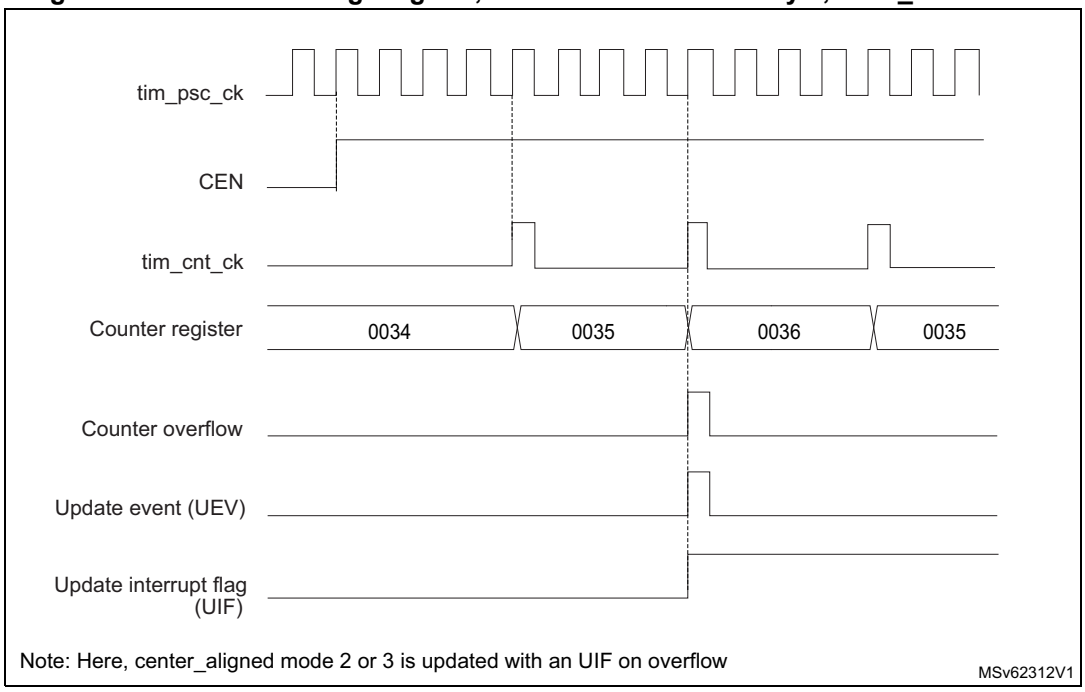


Figure 465. Counter timing diagram, internal clock divided by N

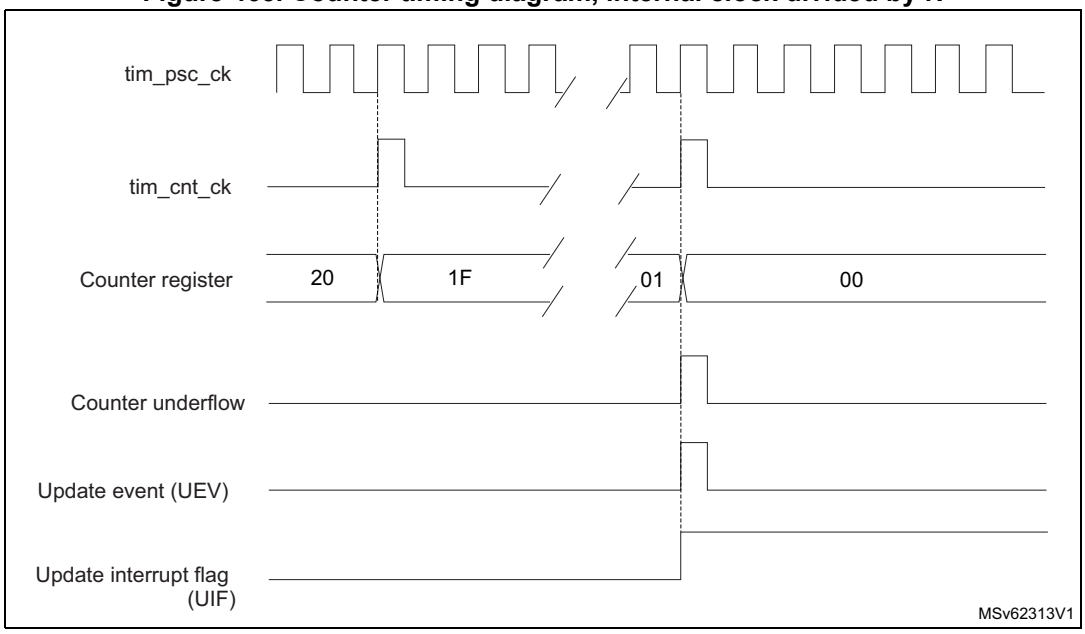


Figure 466. Counter timing diagram, update event with ARPE = 1 (counter underflow)

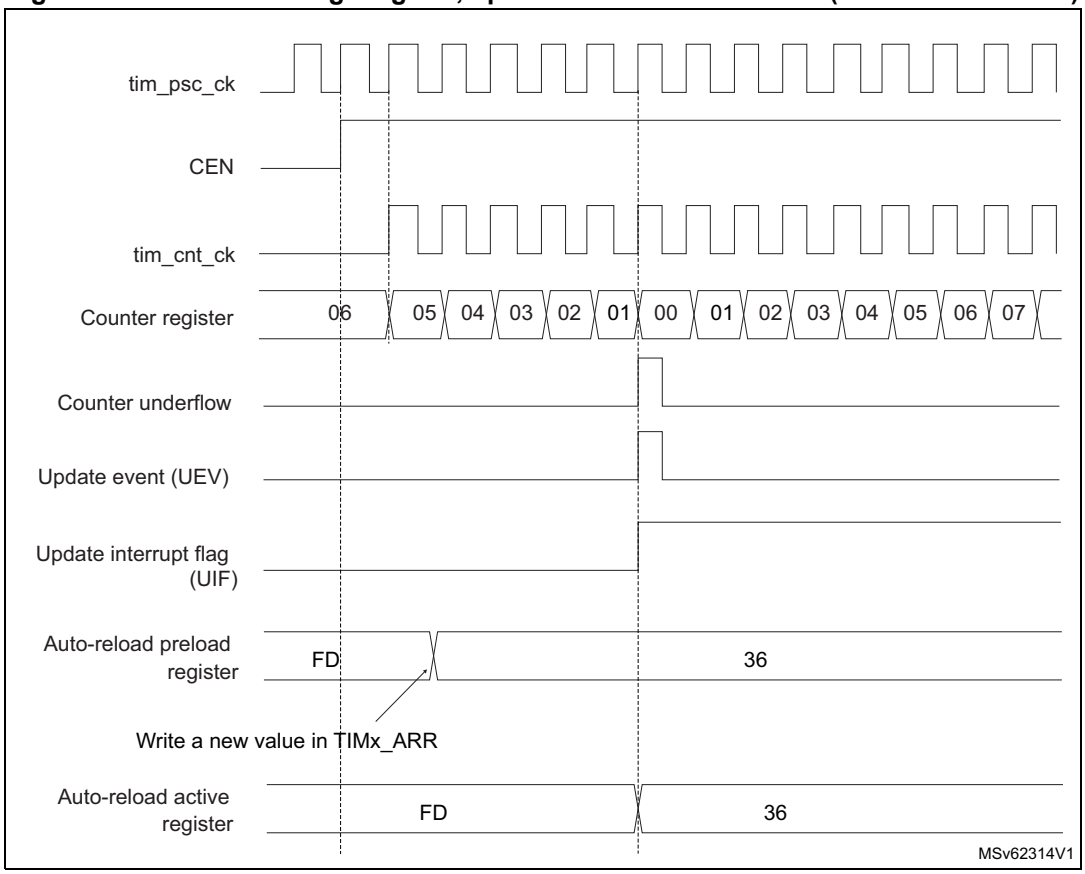
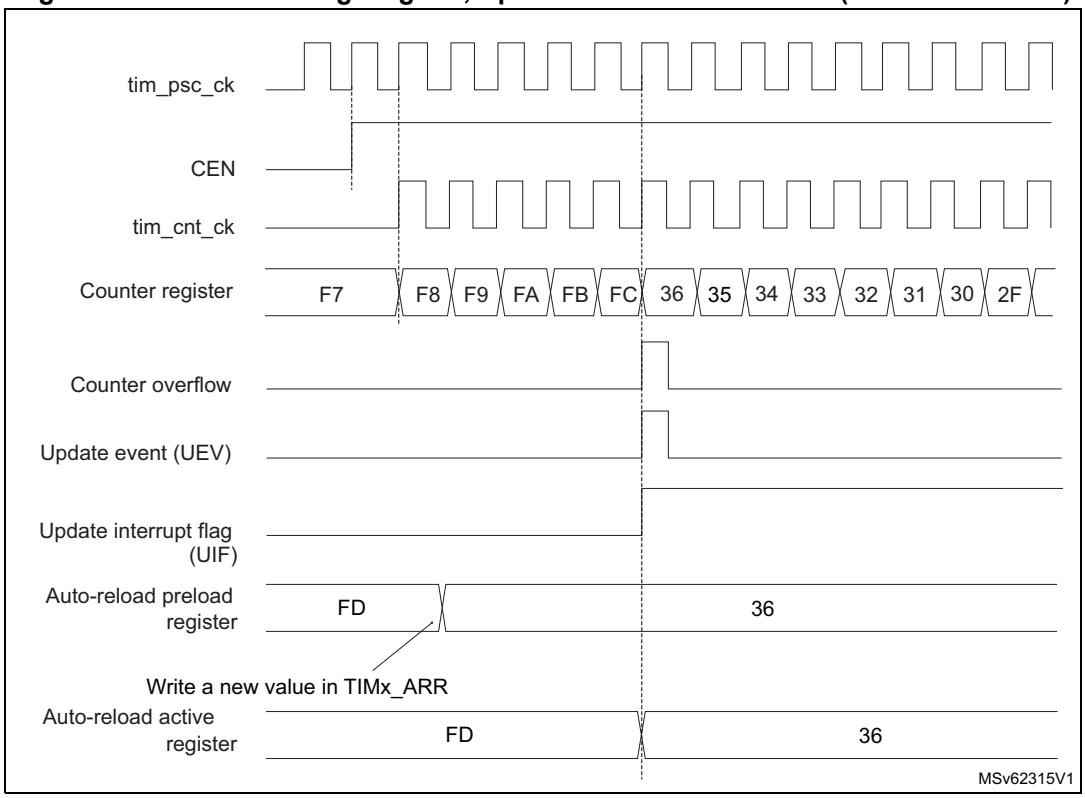


Figure 467. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



53.3.5 Repetition counter

Section 53.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

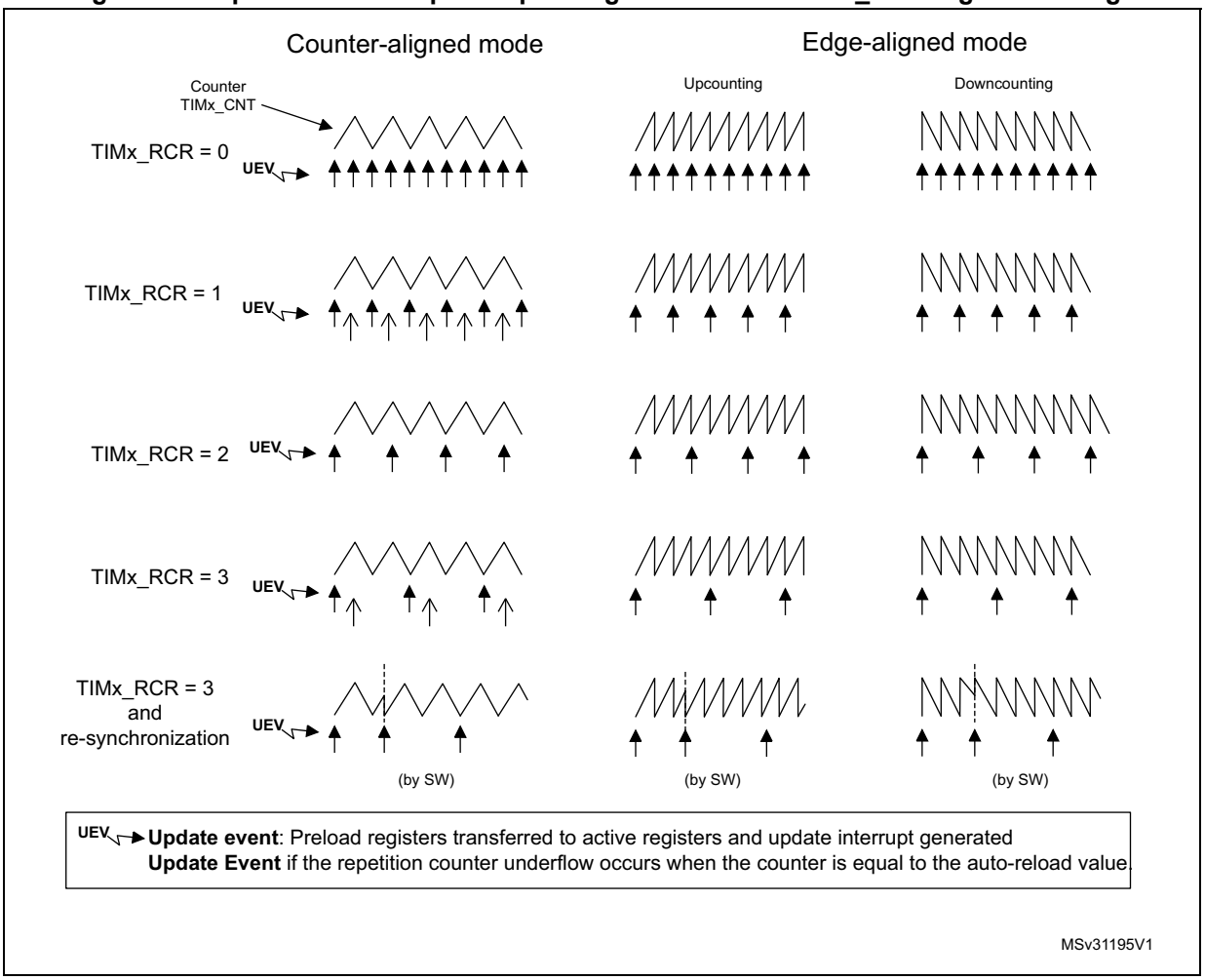
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 468*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 468. Update rate examples depending on mode and TIMx_RCR register settings



53.3.6 External trigger input

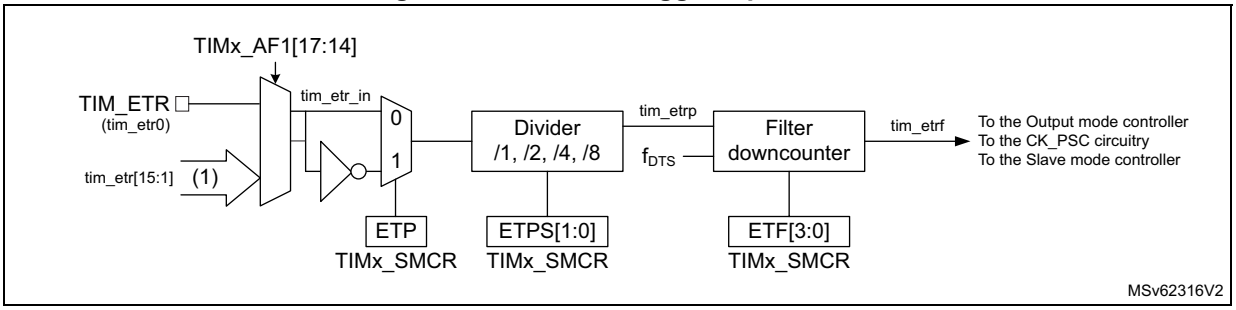
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 53.3.7*)
- trigger for the slave mode (see *Section 53.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 53.3.9*)

Figure 469 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 469. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

53.3.7 Clock selection

The counter clock can be provided by the following clock sources:

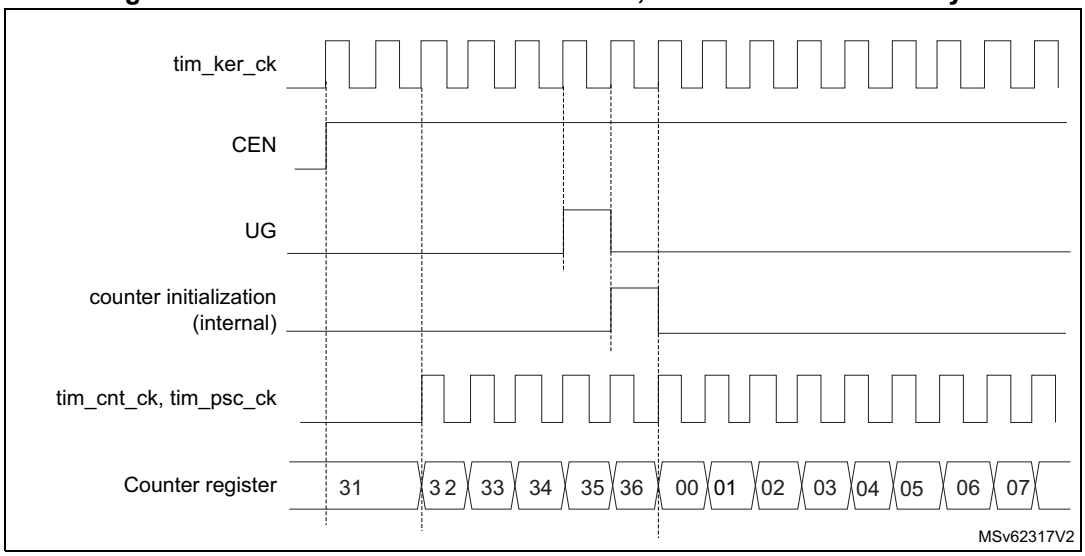
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 470 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

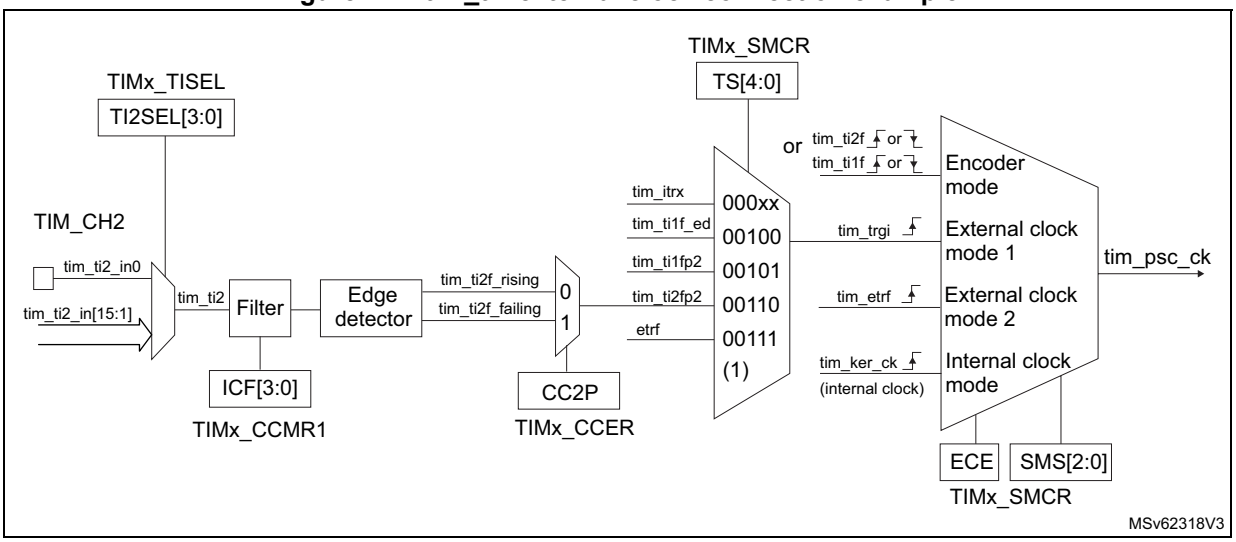
Figure 470. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 471. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

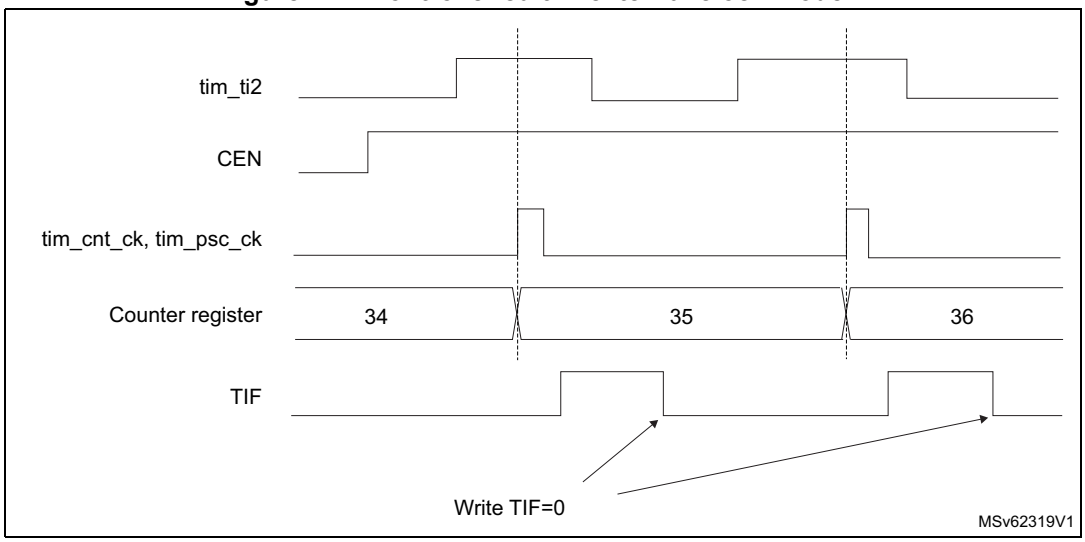
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 472. Control circuit in external clock mode 1



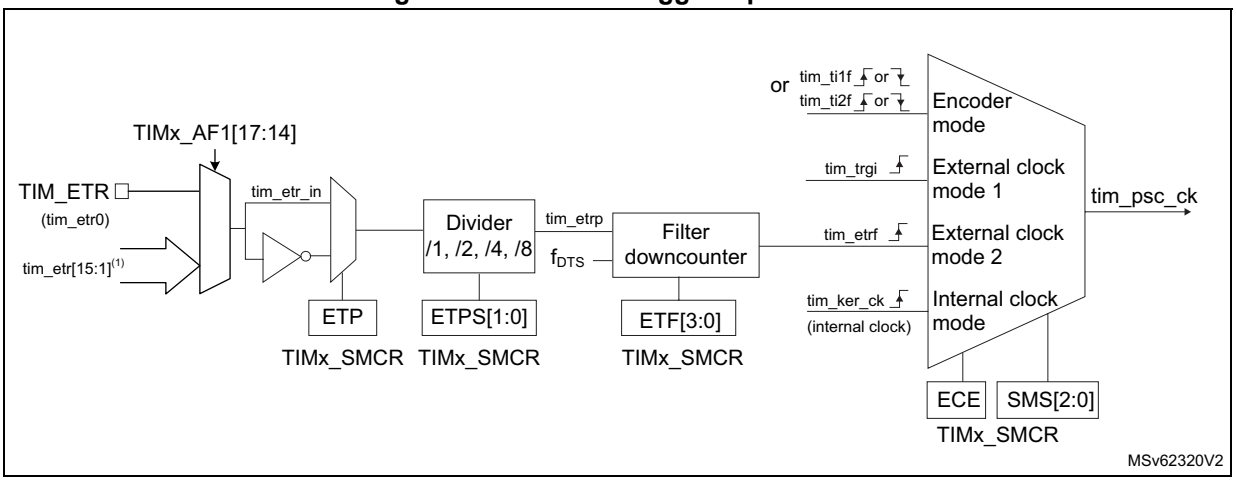
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 473* gives an overview of the external trigger input block.

Figure 473. External trigger input block



1. Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals*.

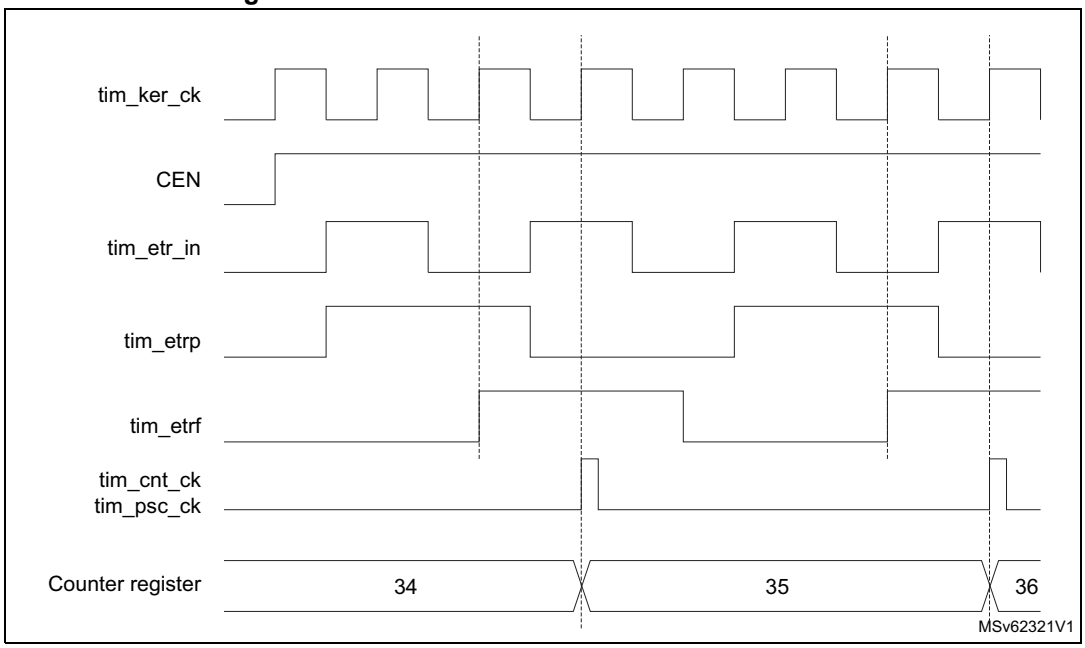
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 474. Control circuit in external clock mode 2



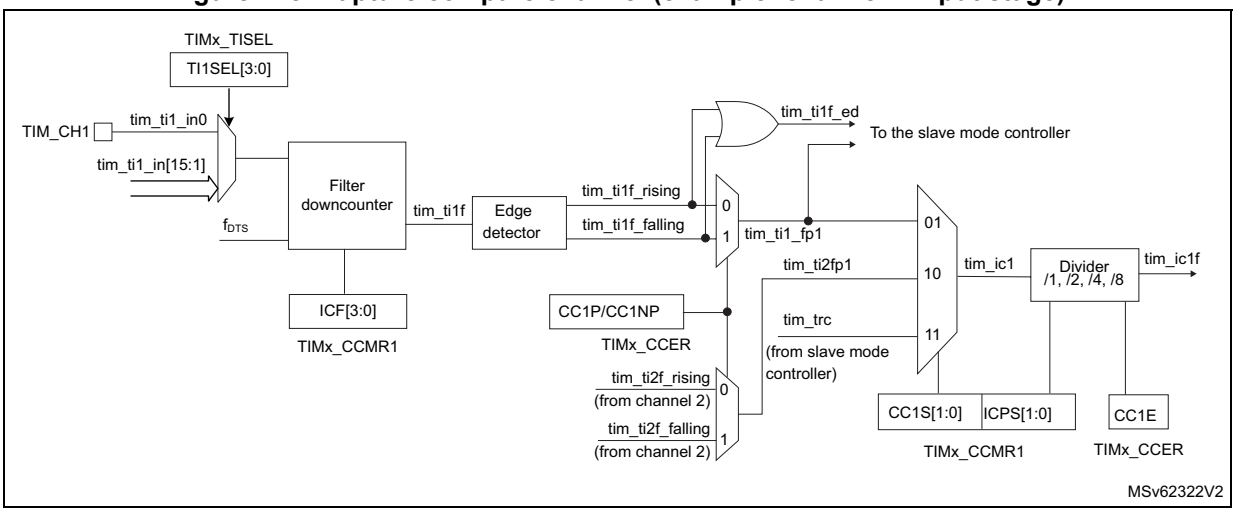
53.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 475 to *Figure 478* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 475. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 476. Capture/compare channel 1 main circuit

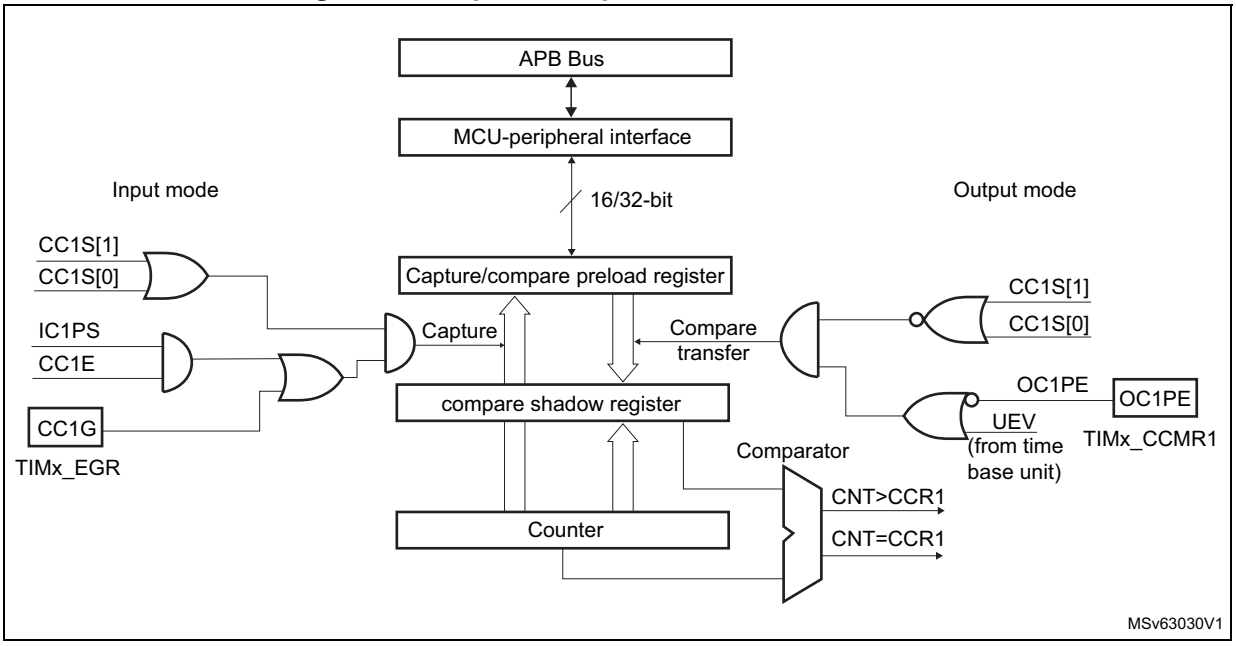
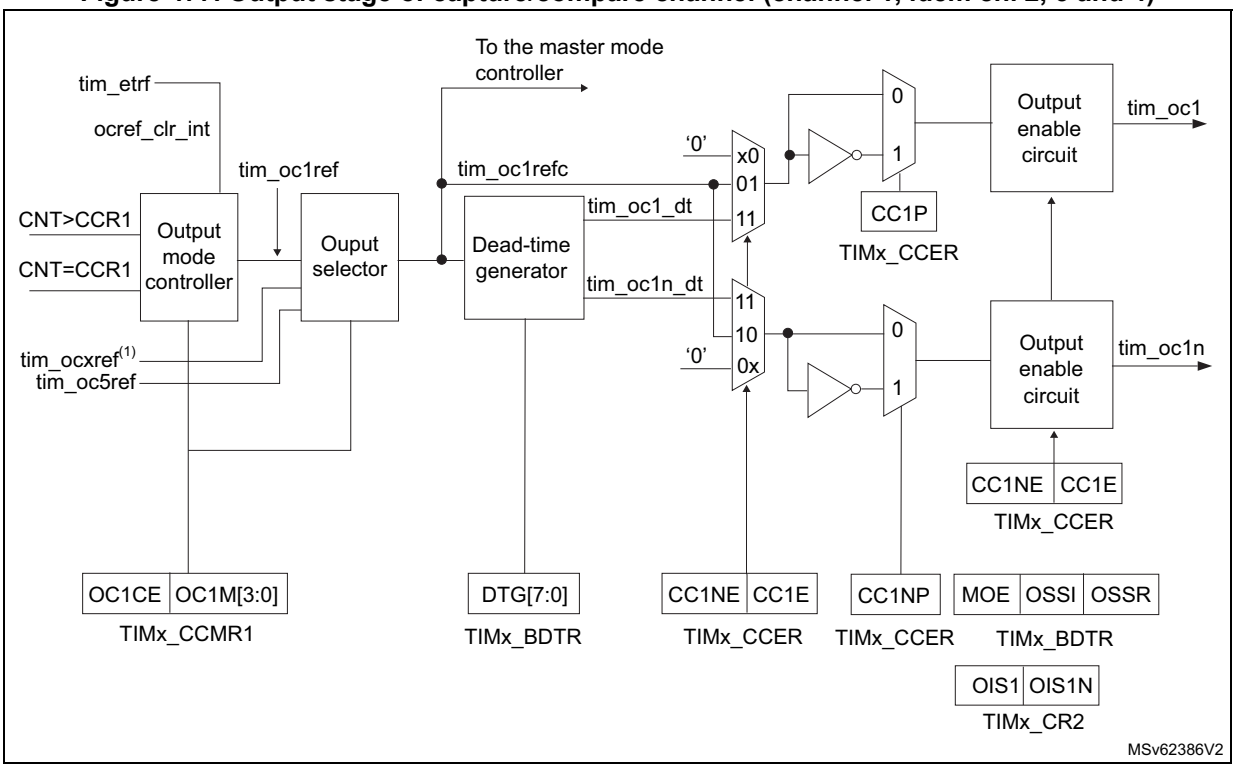
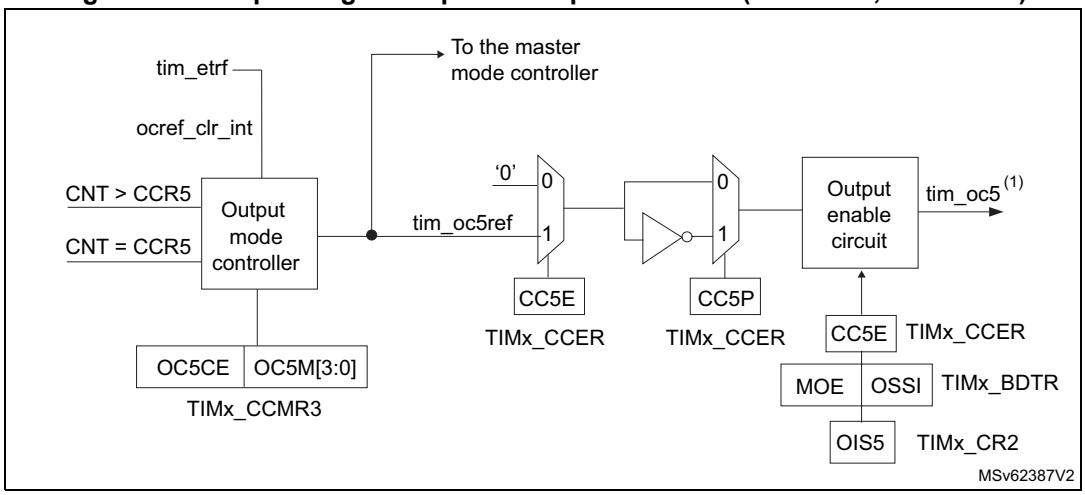


Figure 477. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 478. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

53.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level

have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

53.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

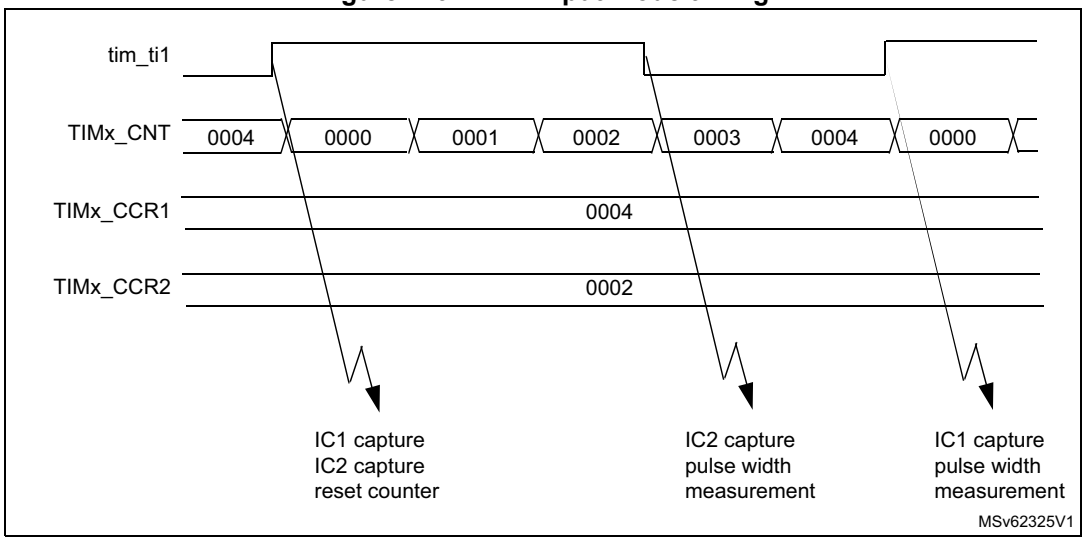
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 479. PWM input mode timing



53.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

53.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

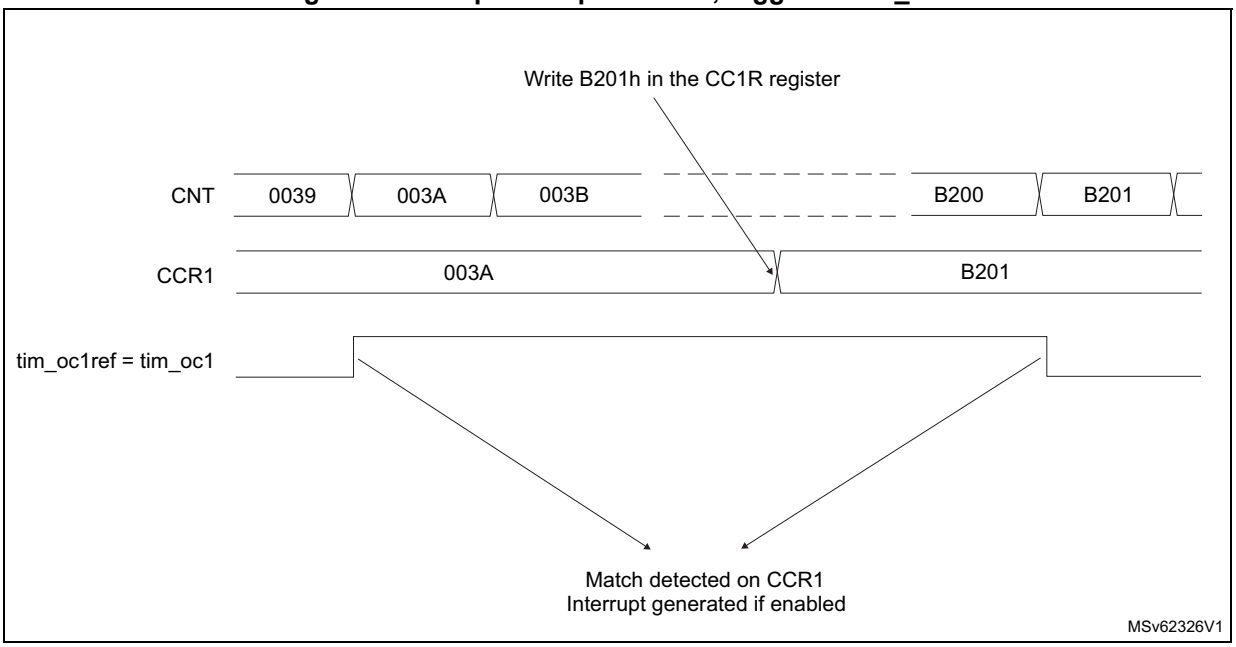
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 480*.

Figure 480. Output compare mode, toggle on tim_oc1



53.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

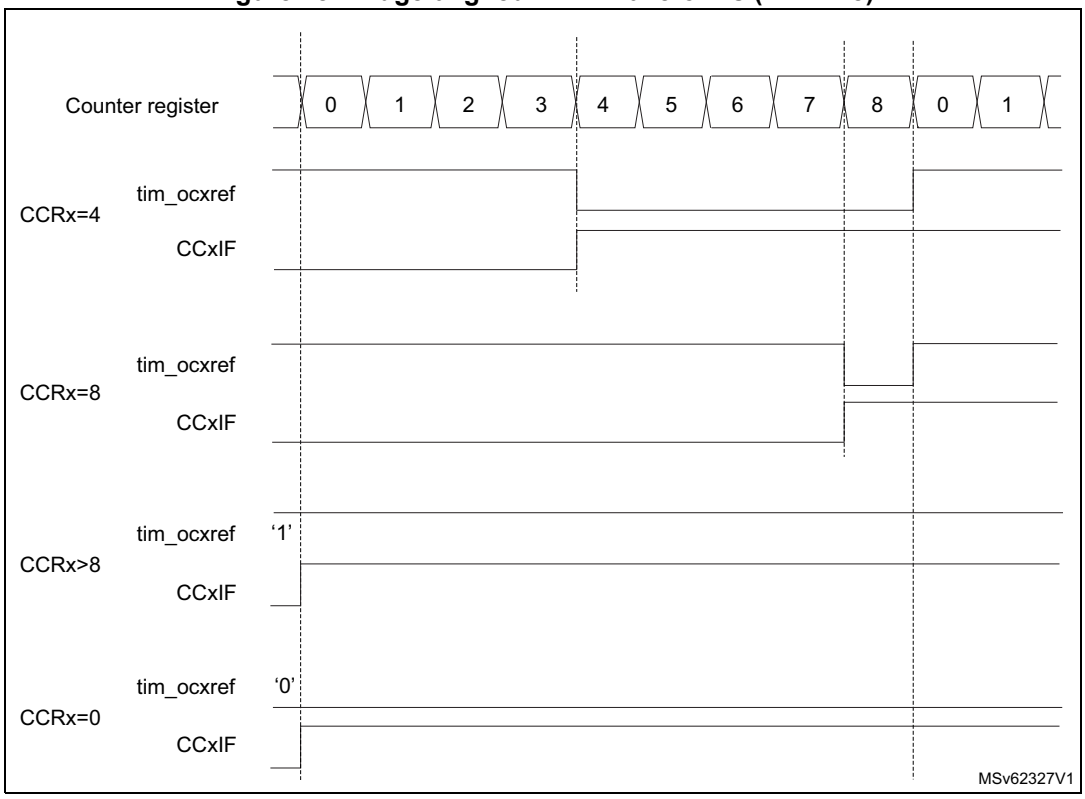
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 481* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 481. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

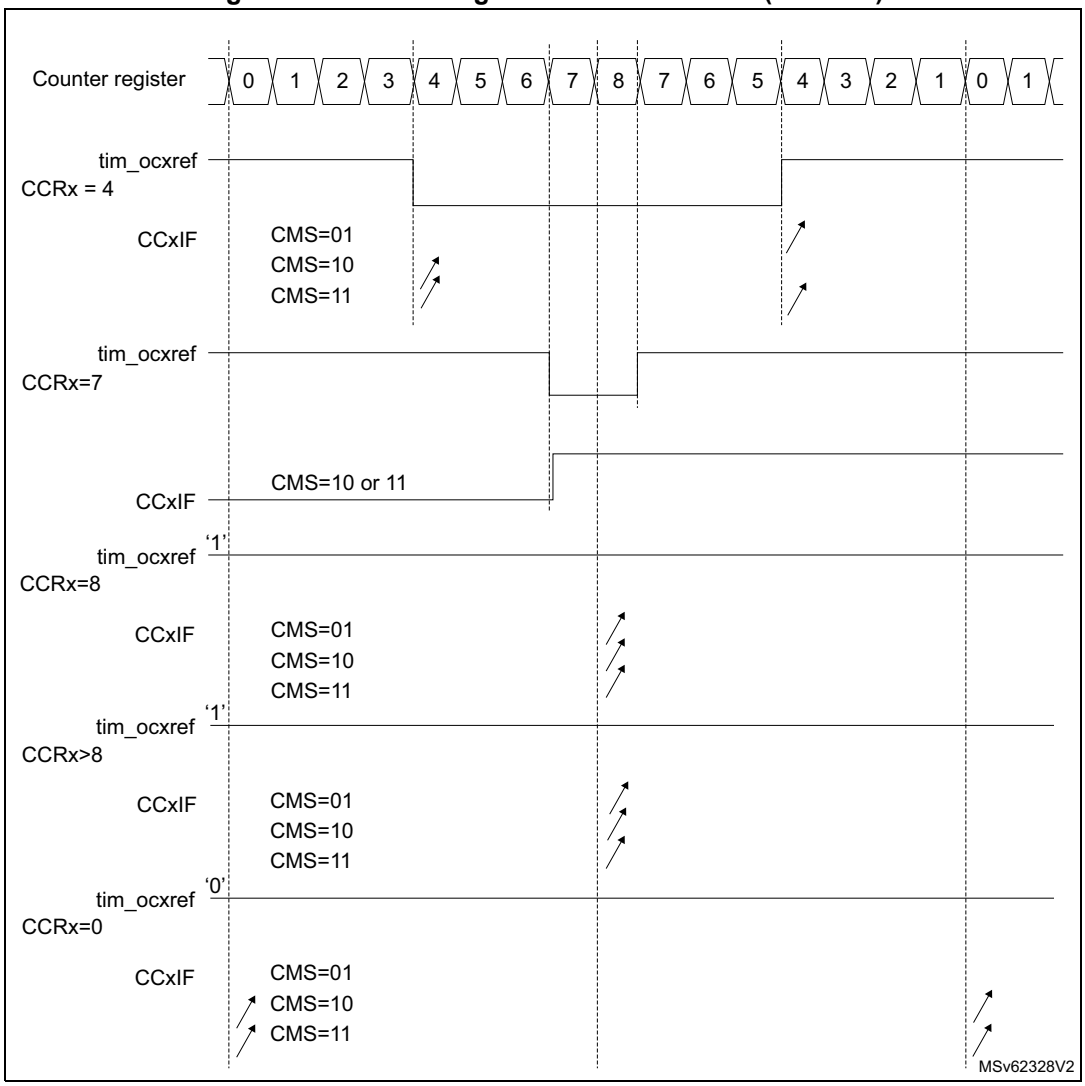
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 482 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 482. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

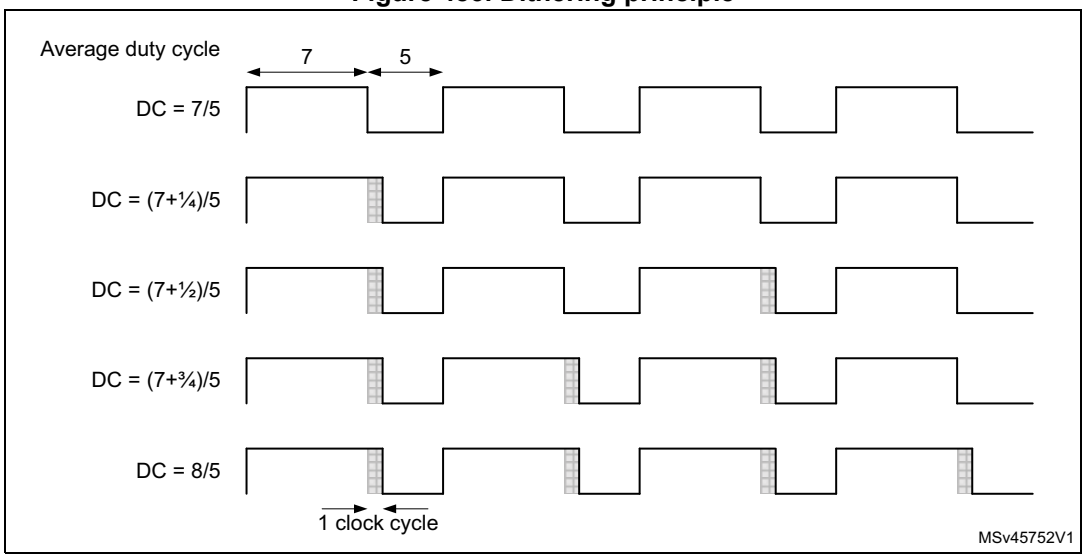
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 483* presents the dithering principle applied to four consecutive PWM cycles.

Figure 483. Dithering principle

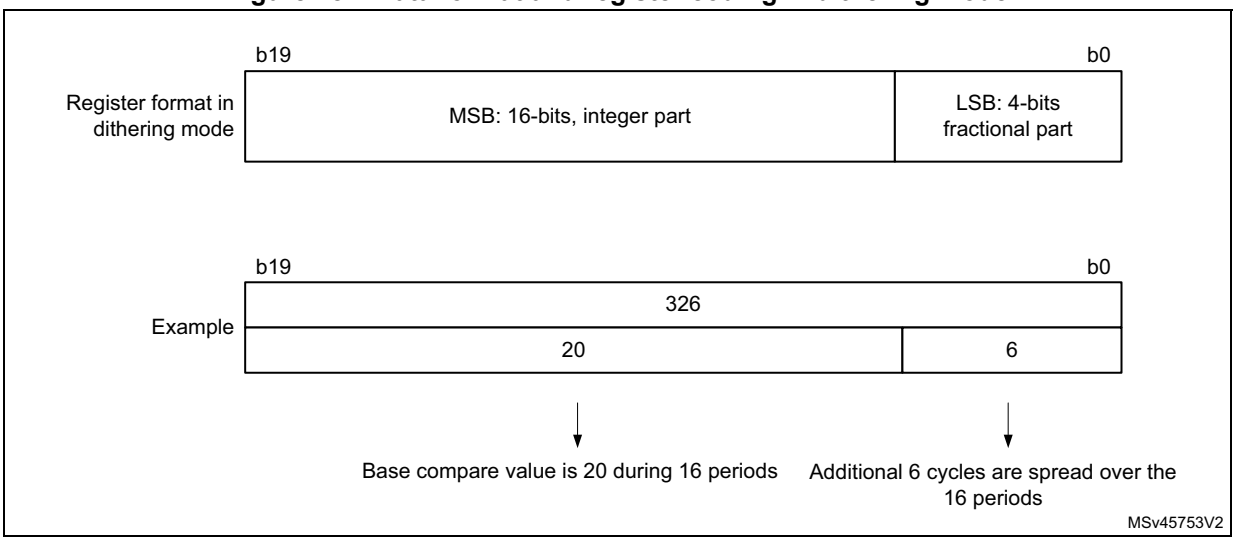


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 484* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 484. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

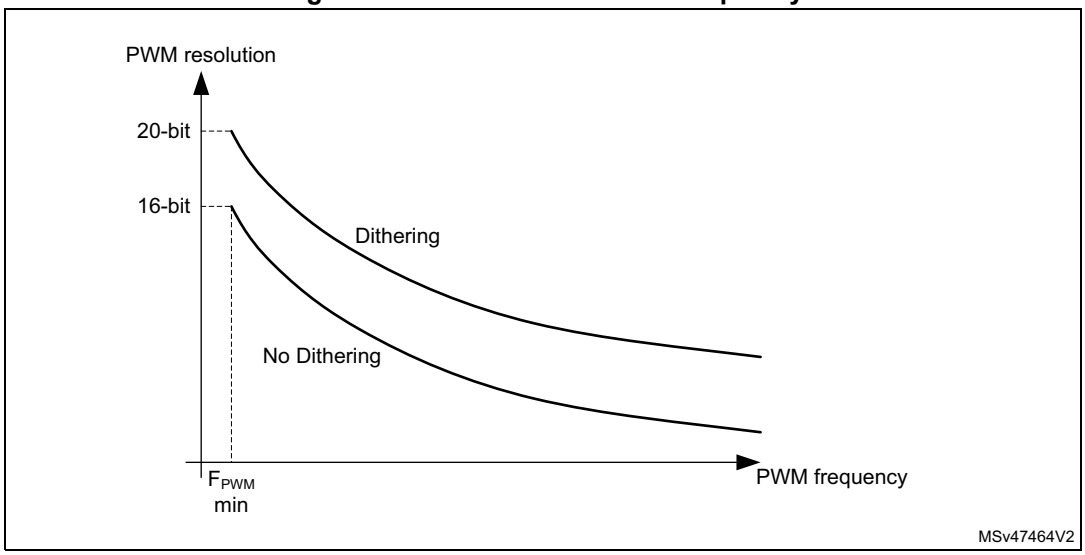
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

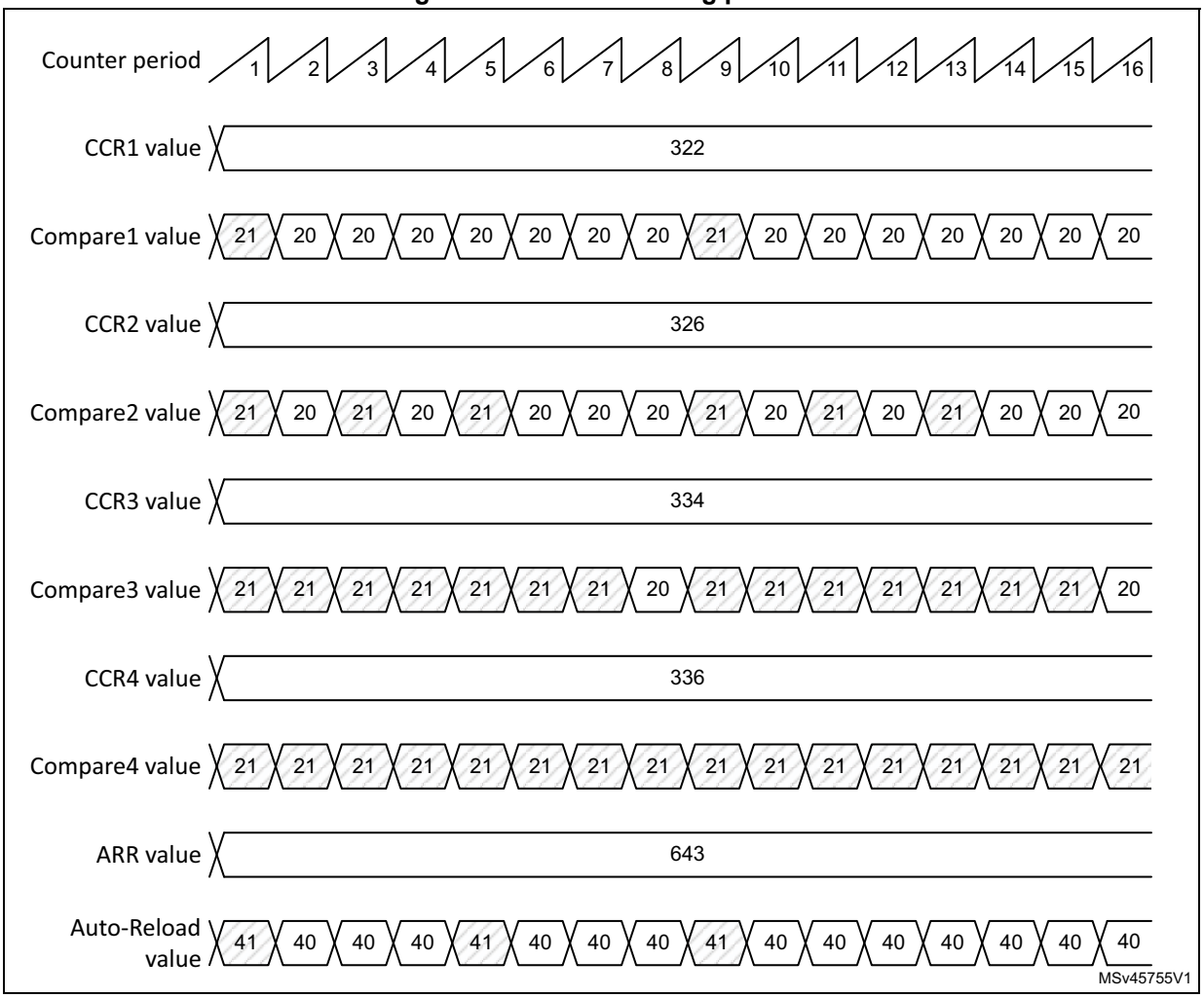
As shown on *Figure 485*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 485. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 486*.

Figure 486. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 529*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 529. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 529. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 487*.

Figure 487. Dithering effect on duty cycle in center-aligned PWM mode

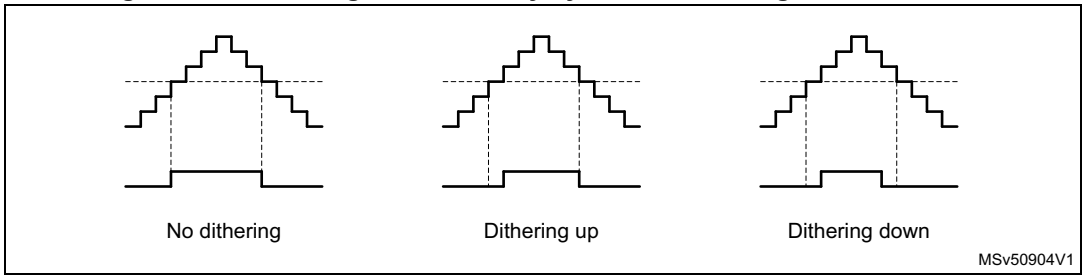


Table 530 shows how the dithering pattern is added in center-aligned PWM mode.

Table 530. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 530. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

53.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

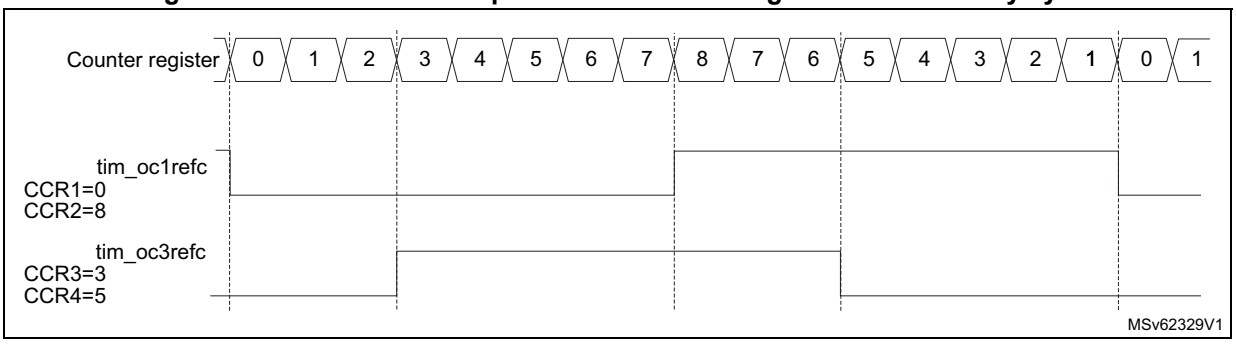
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 488 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 488. Generation of 2 phase-shifted PWM signals with 50% duty cycle



53.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

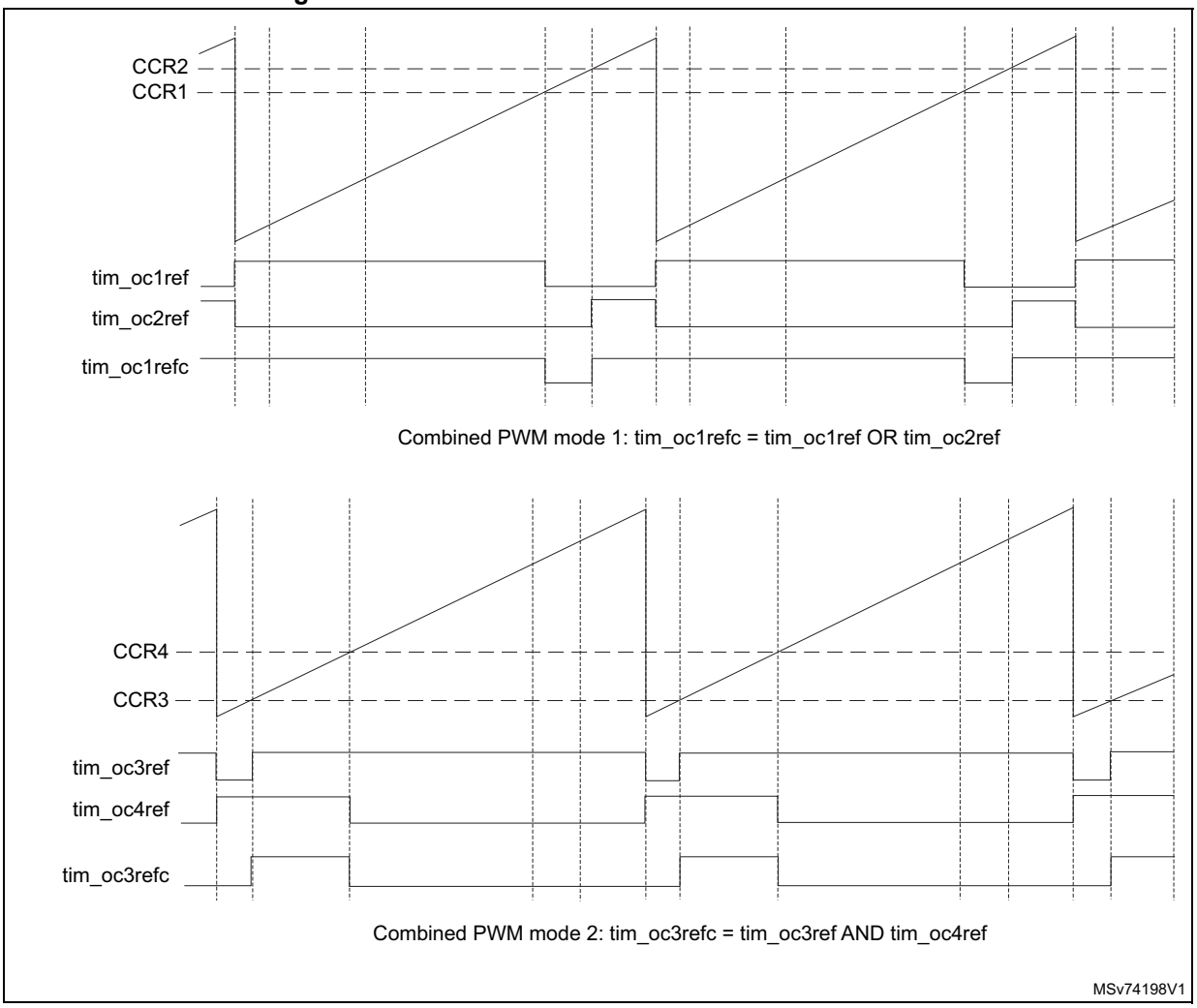
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 489 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 489. Combined PWM mode on channel 1 and 3



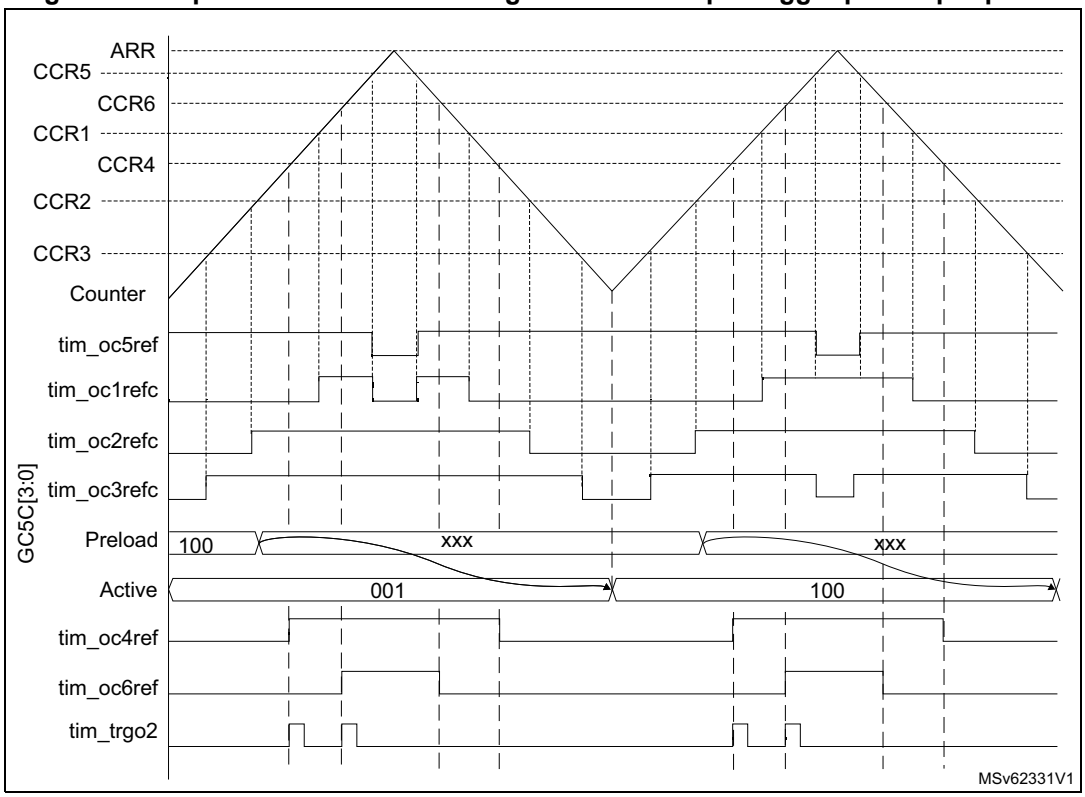
53.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 490. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 53.3.31: ADC triggers* for more details.

53.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 538: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

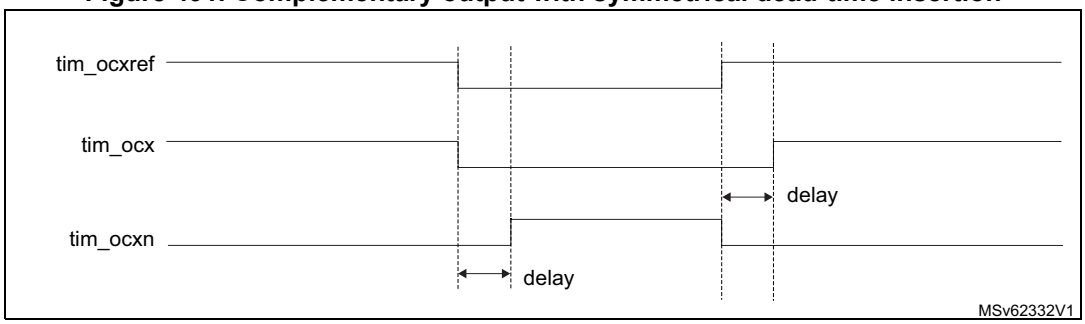
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 491. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 492*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 492. Asymmetrical deadtime

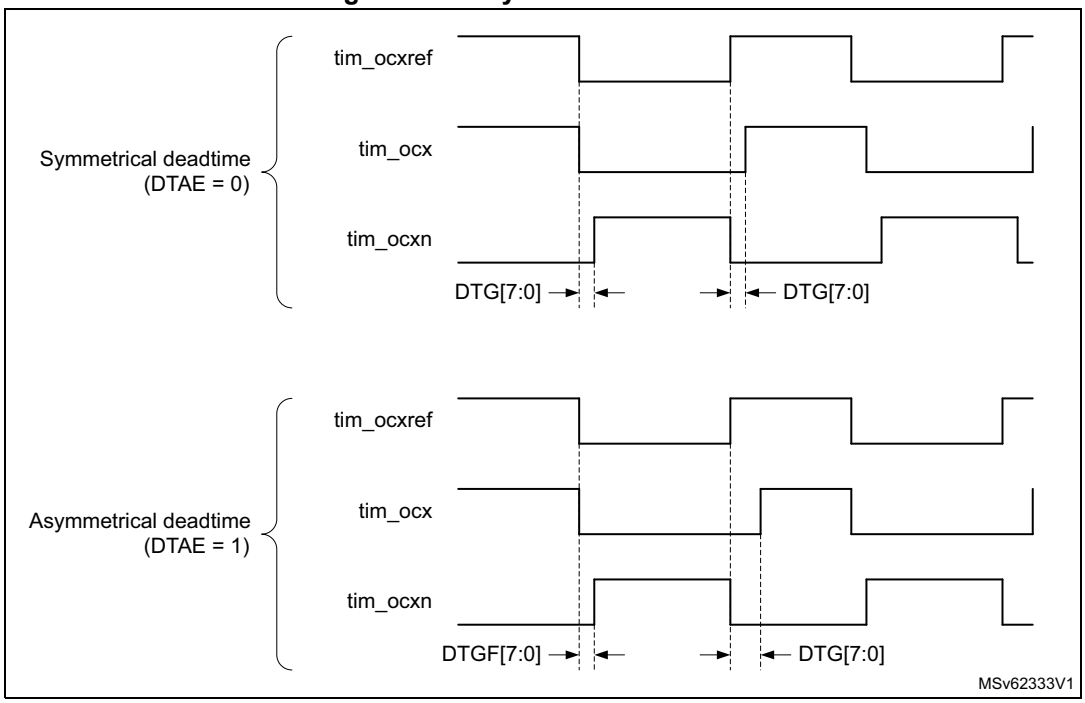


Figure 493. Dead-time waveforms with delay greater than the negative pulse

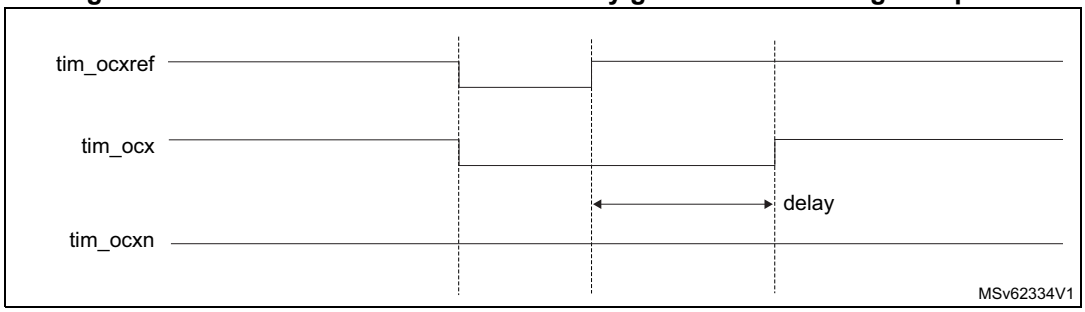
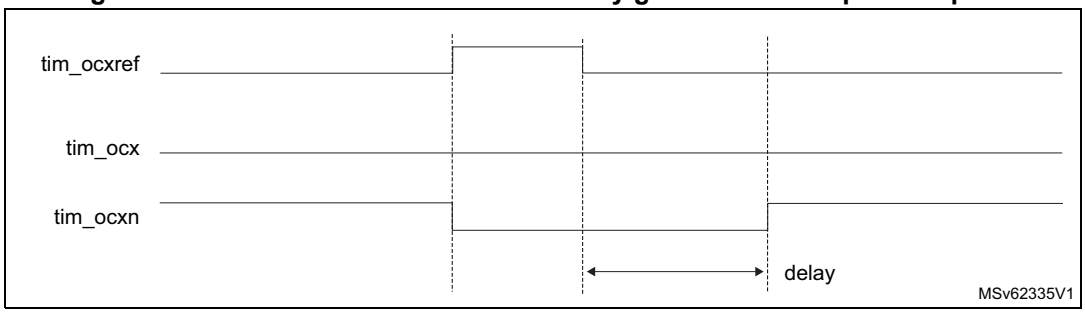


Figure 494. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 53.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

53.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 538: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

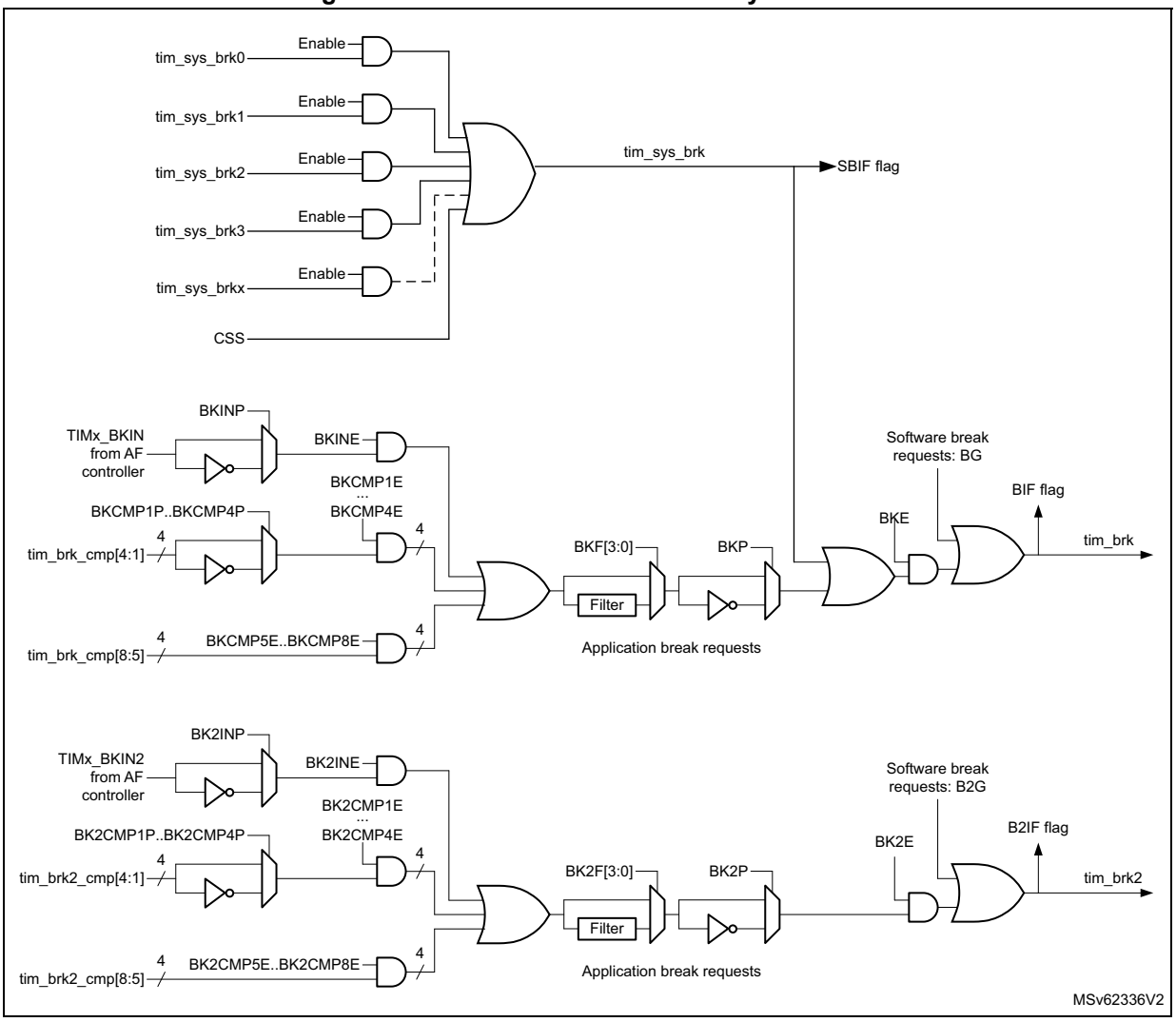
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 495* below.

Figure 495. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

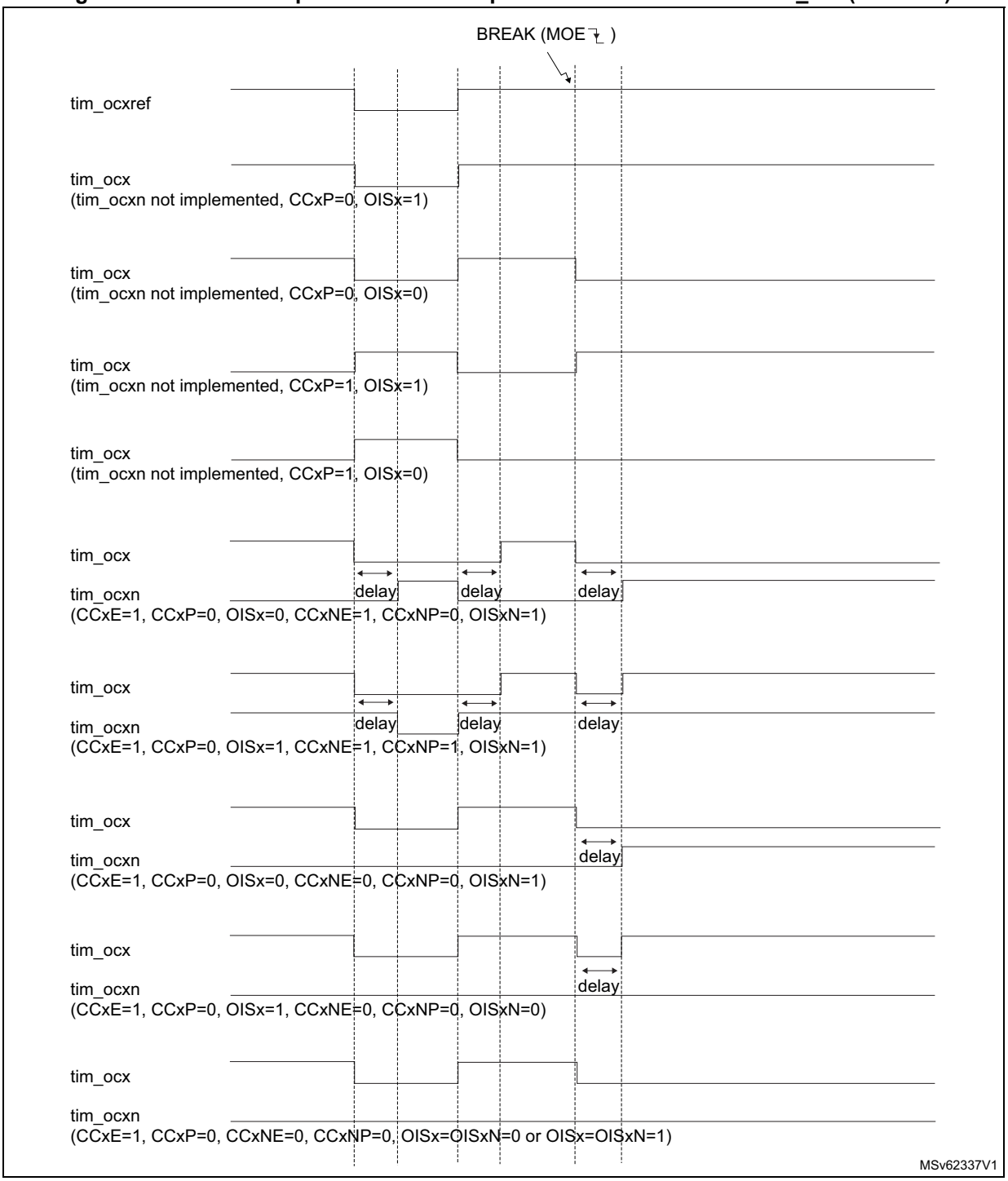
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 53.6.20*. The LOCK bits can be written only once after an MCU reset.

Figure 496 shows an example of behavior of the outputs in response to a break.

Figure 496. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 531*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 531. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 497 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 497. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

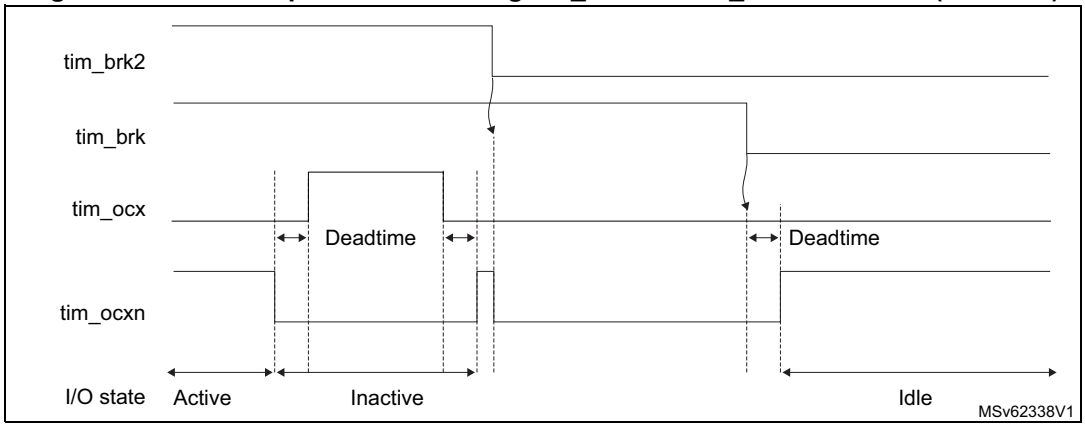
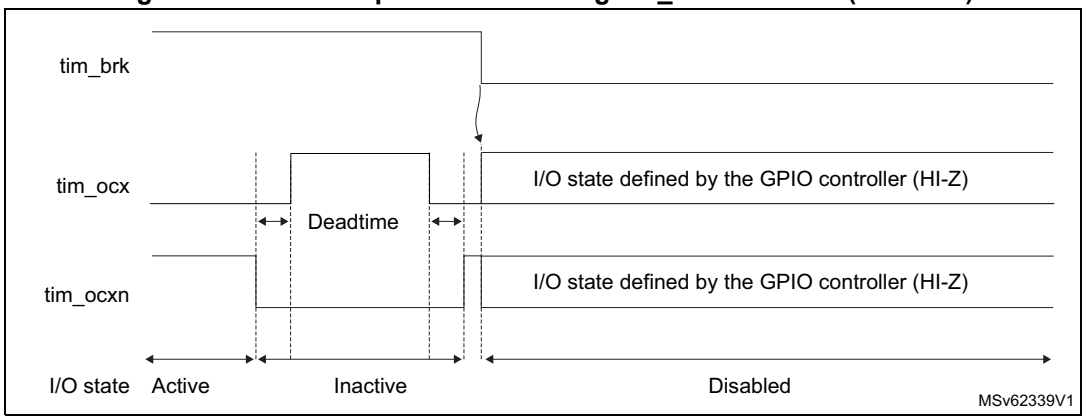


Figure 498. PWM output state following tim_brk assertion (OSSI = 0)



53.3.19 Bidirectional break inputs

The TIM1/TIM8 feature bidirectional break I/Os, as represented on *Figure 499*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 532*).

Table 532. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

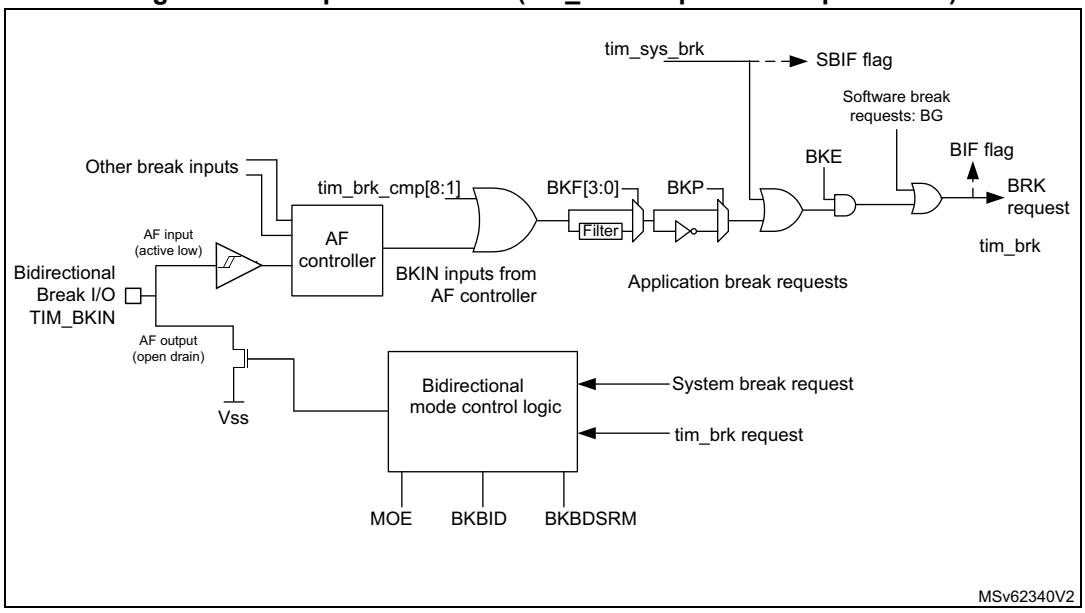
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 499. Output redirection (tim_brk2 request not represented)



53.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM

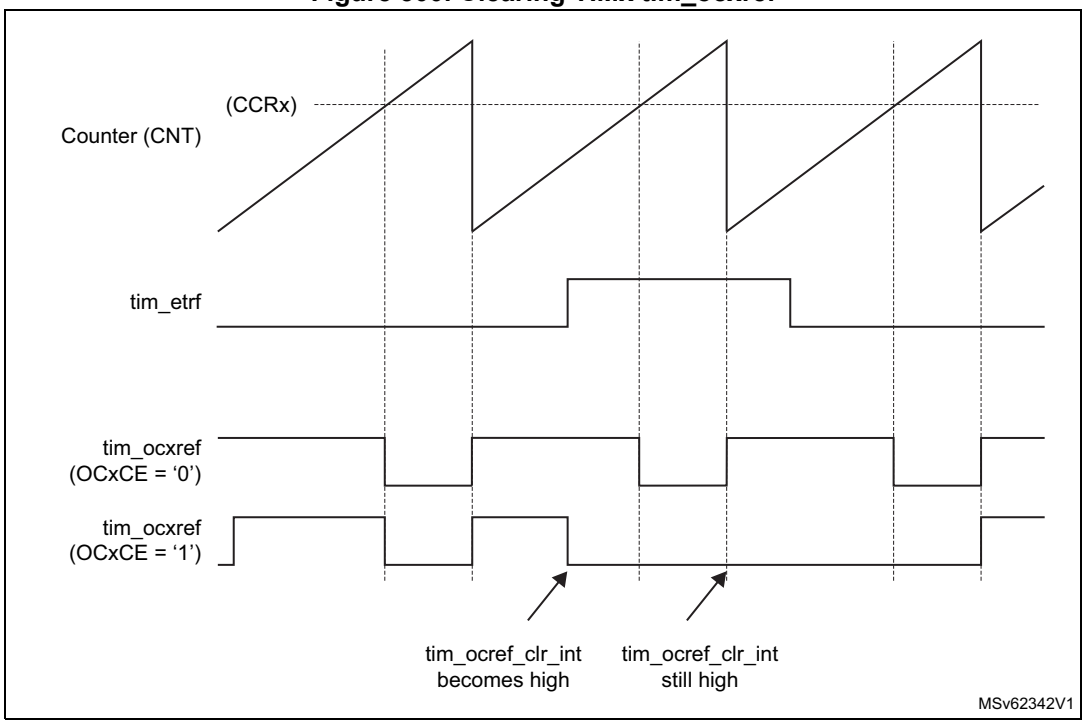
cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 500 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 500. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

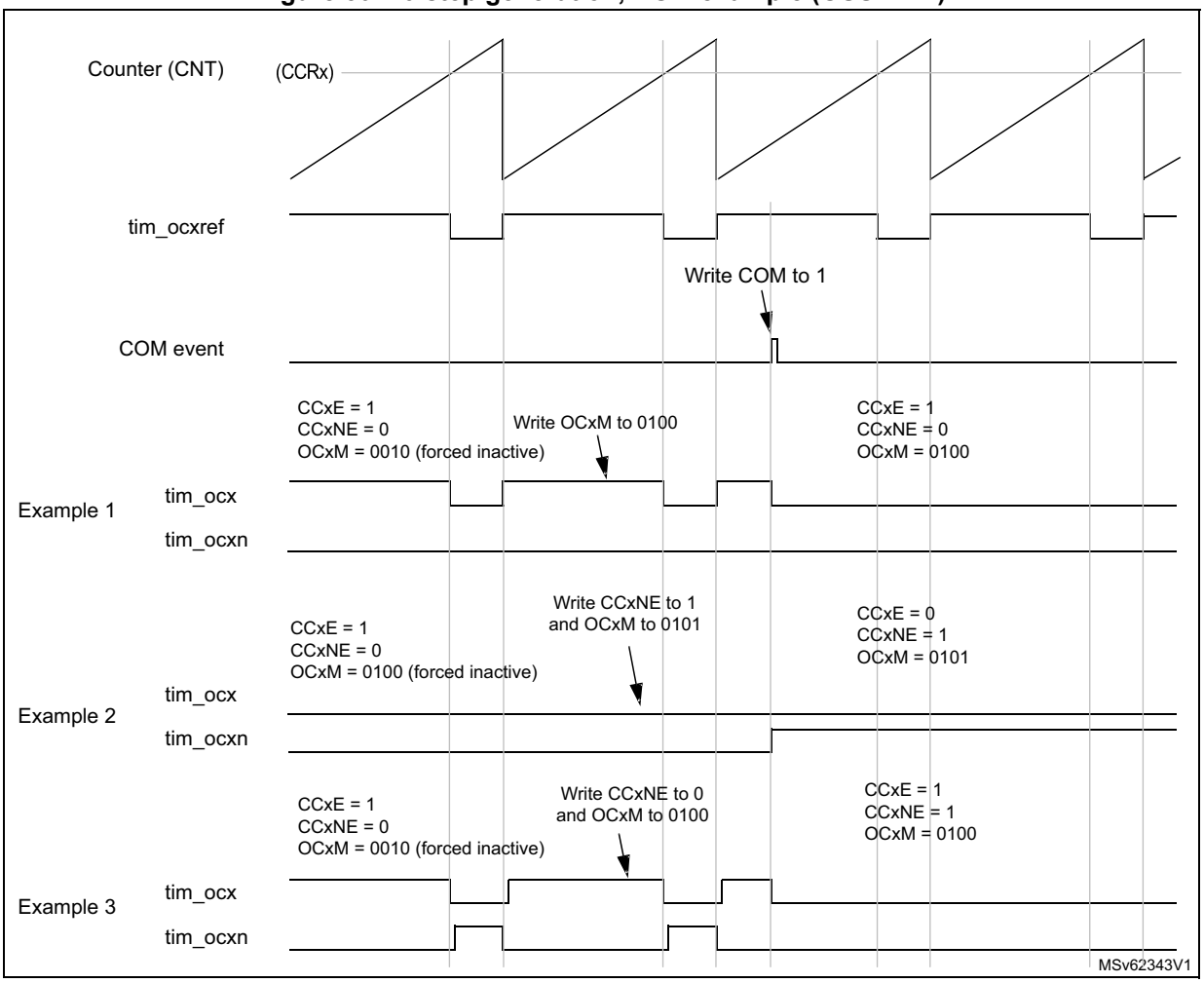
53.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 501 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 501. 6-step generation, COM example (OSSR = 1)



53.3.22 One-pulse mode

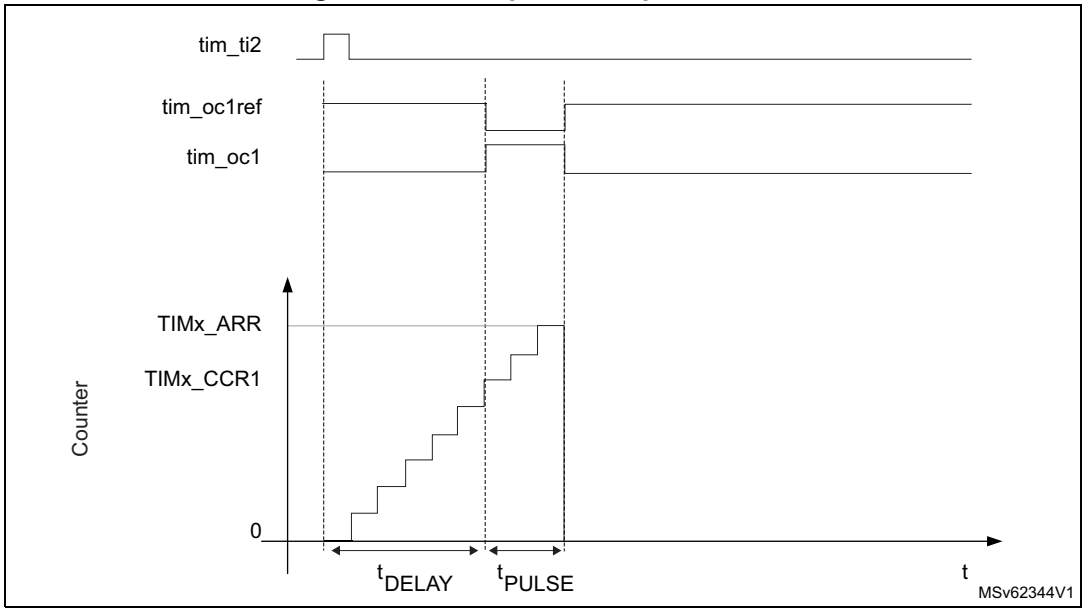
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 502. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the

TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

53.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 53.3.22*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

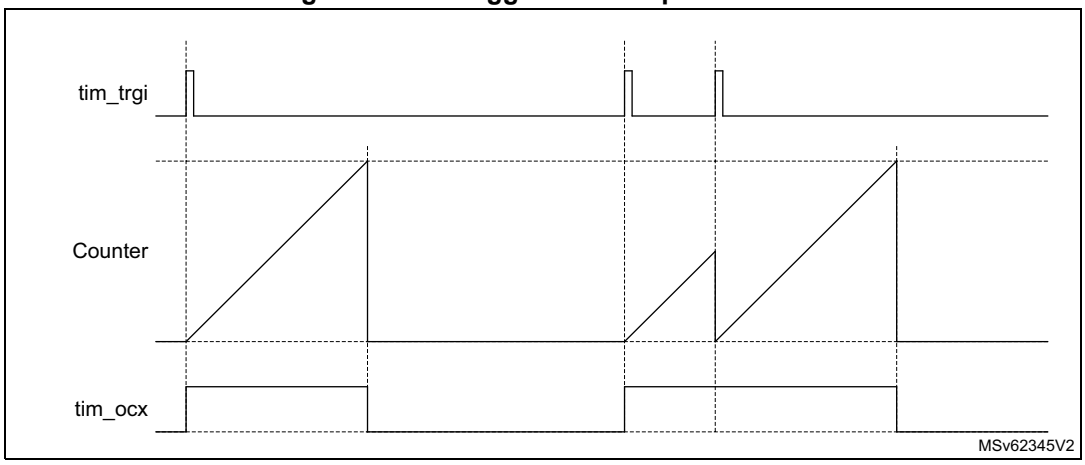
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 503. Retriggerable one-pulse mode



53.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 504*.

Figure 504. Pulse generator circuitry

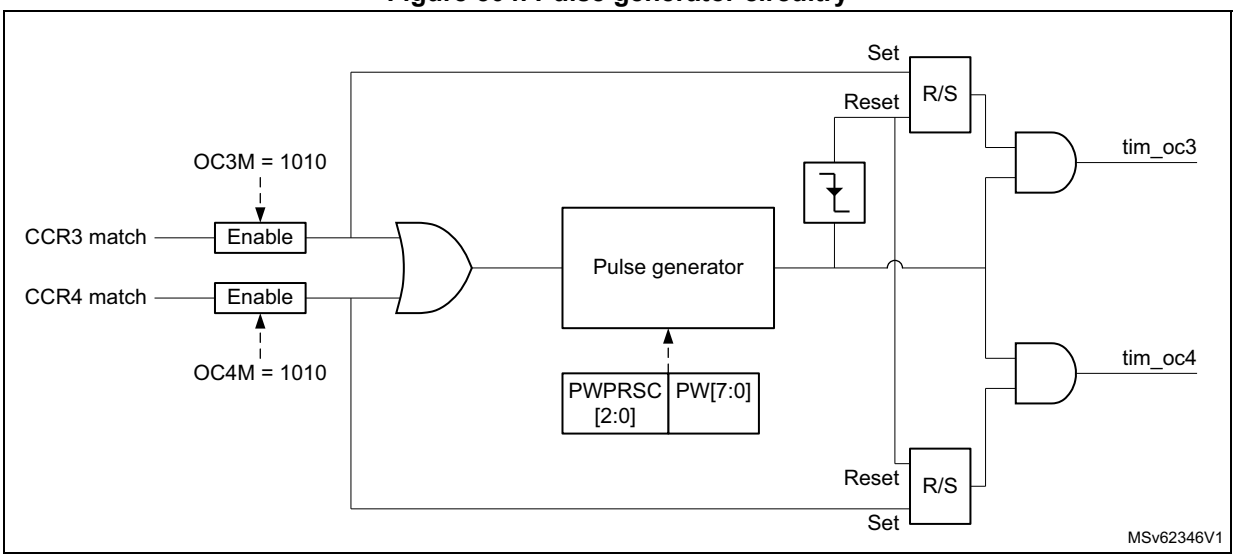
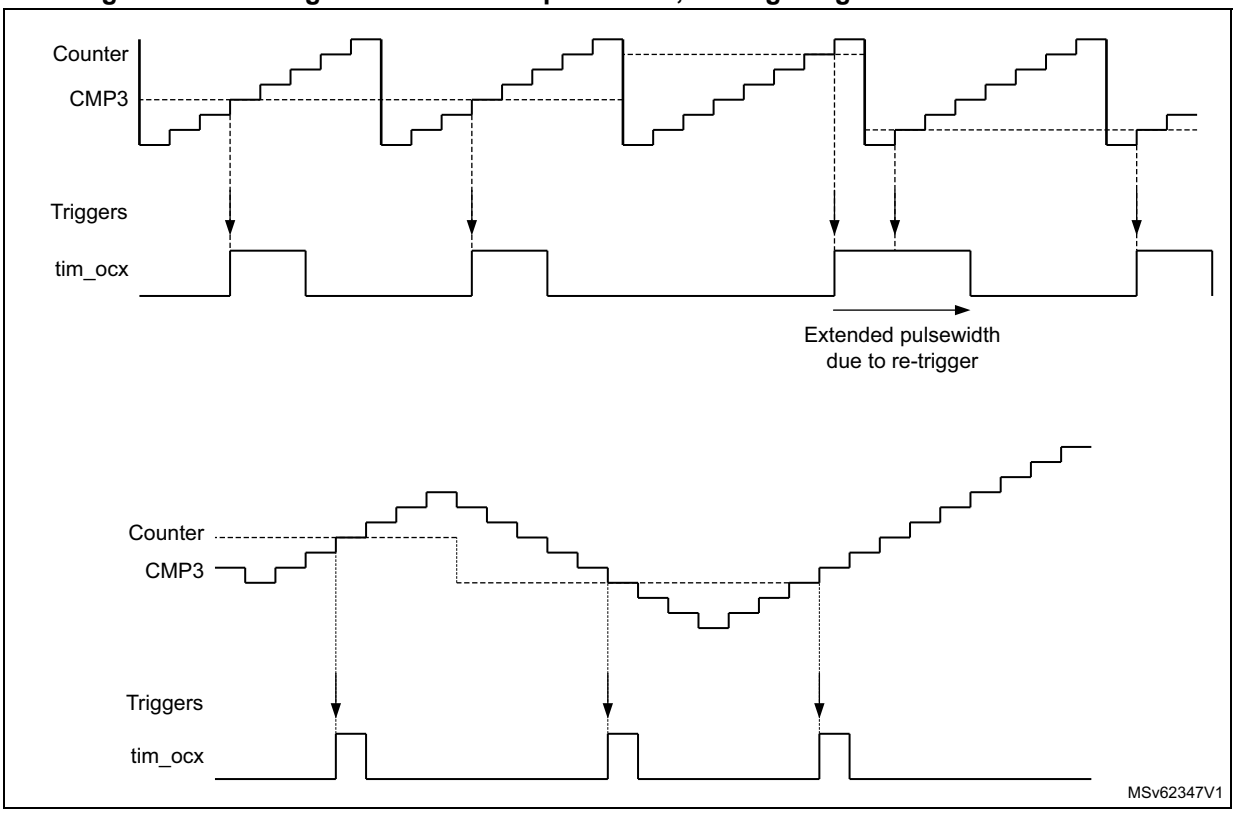


Figure 505 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 505. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

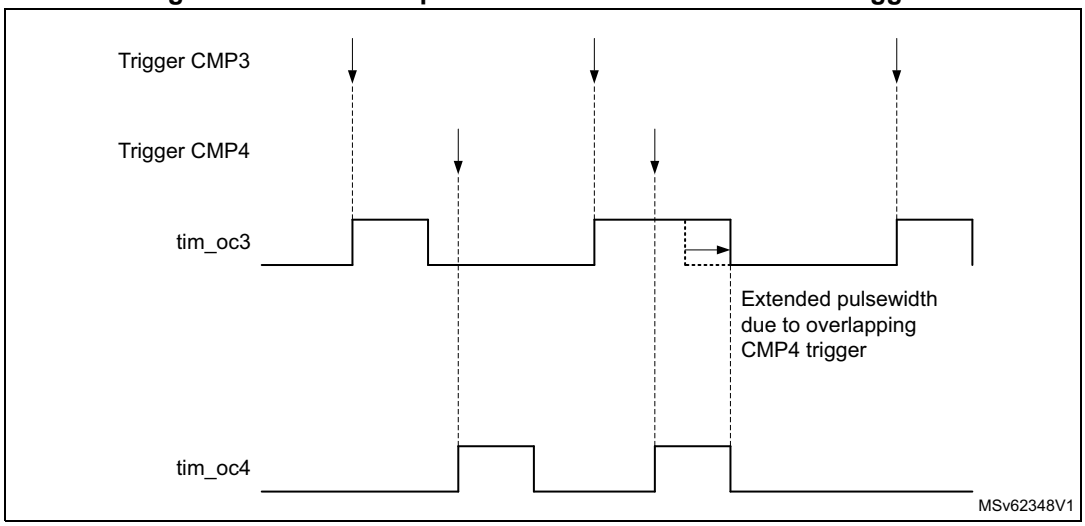
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 506).

Figure 506. Extended pulsewidth in case of concurrent triggers



53.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 533*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 533. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 507 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 507. Example of counter operation in encoder interface mode.

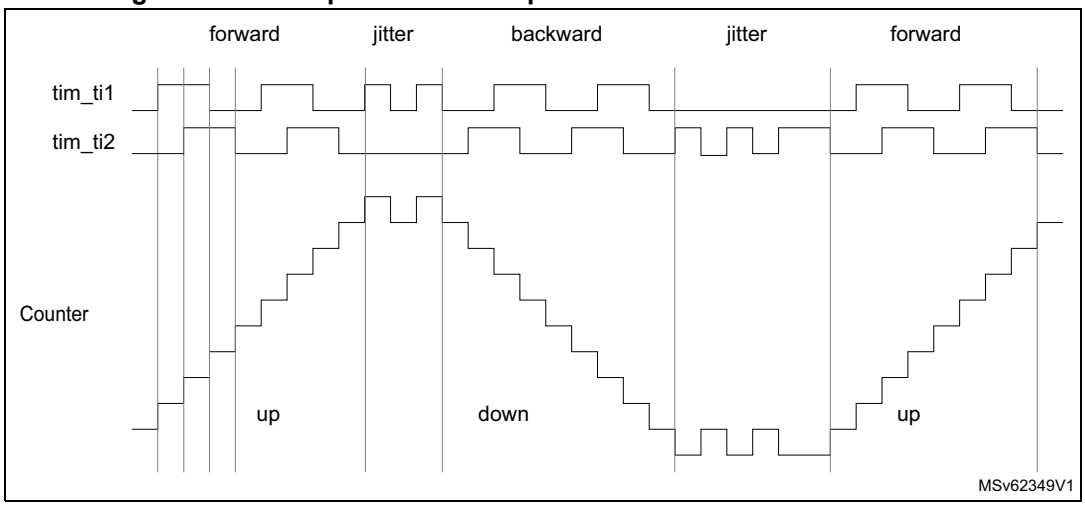


Figure 508 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 508. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

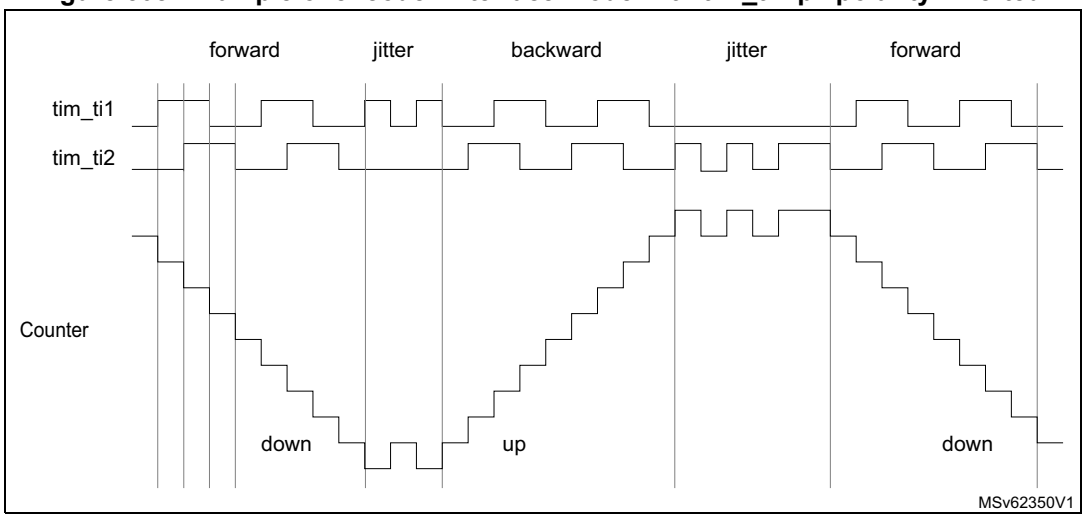
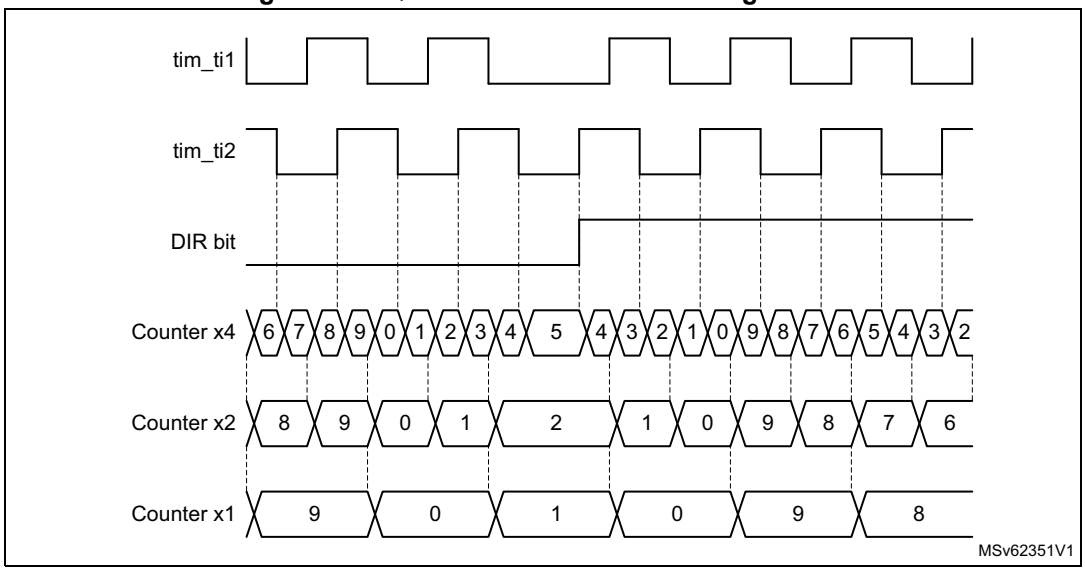


Figure 509 shows the timer counter value during a speed reversal, for various counting modes.

Figure 509. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

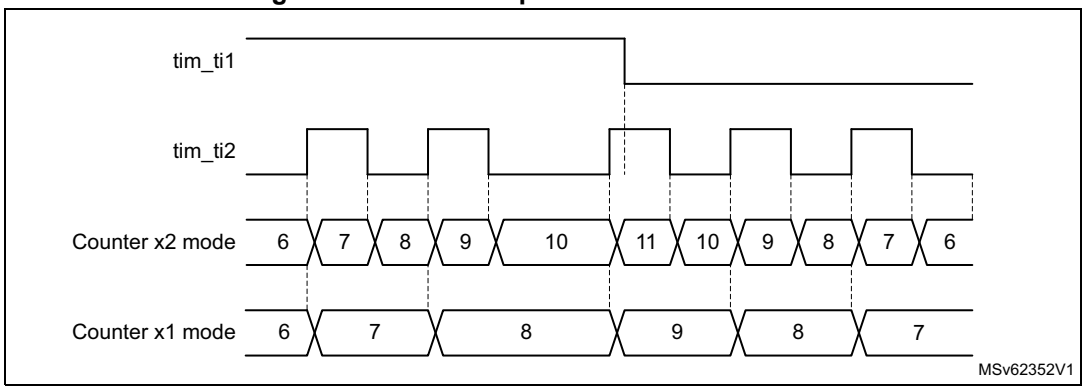
In the clock plus direction mode shown on *Figure 510*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 510. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 511*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 511*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 512*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 511*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 512*).

Figure 511. Directional clock encoder mode (CC1P = CC2P = 0)

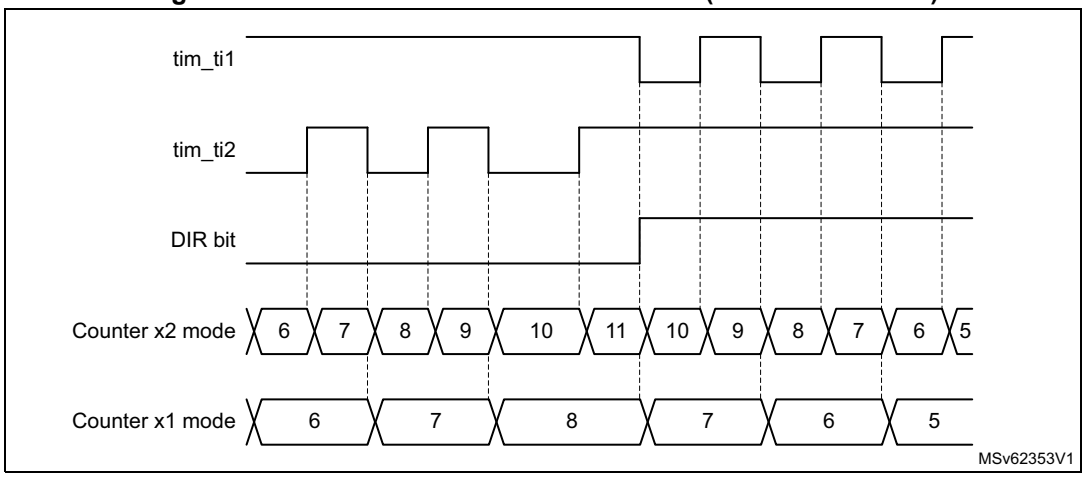


Figure 512. Directional clock encoder mode (CC1P = CC2P = 1)

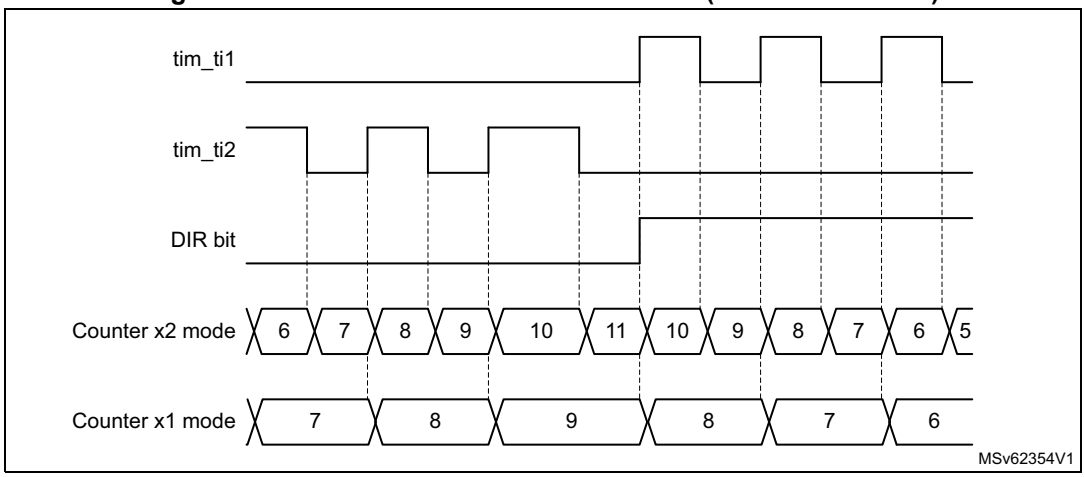


Table 534 here-below details how the directional clock mode operates, for any input transition.

Table 534. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

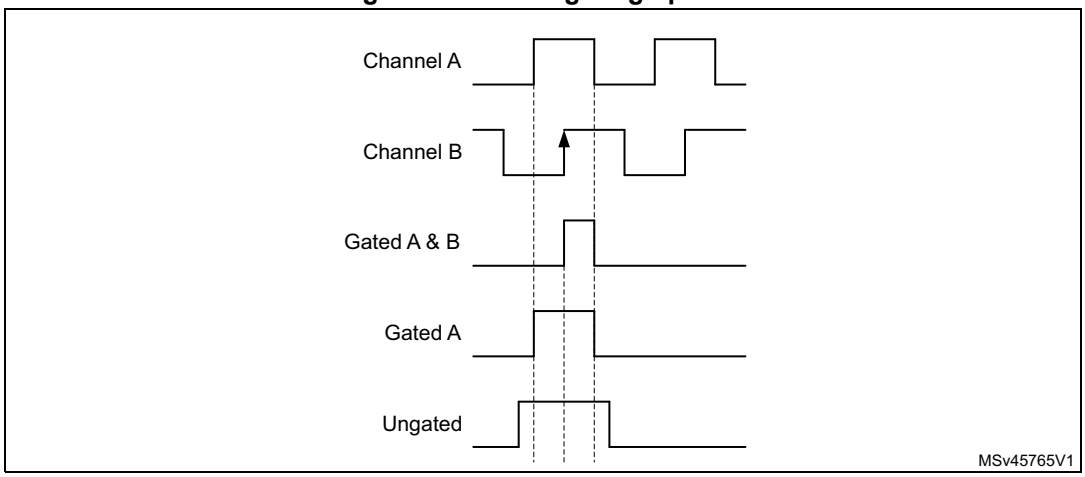
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 513*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

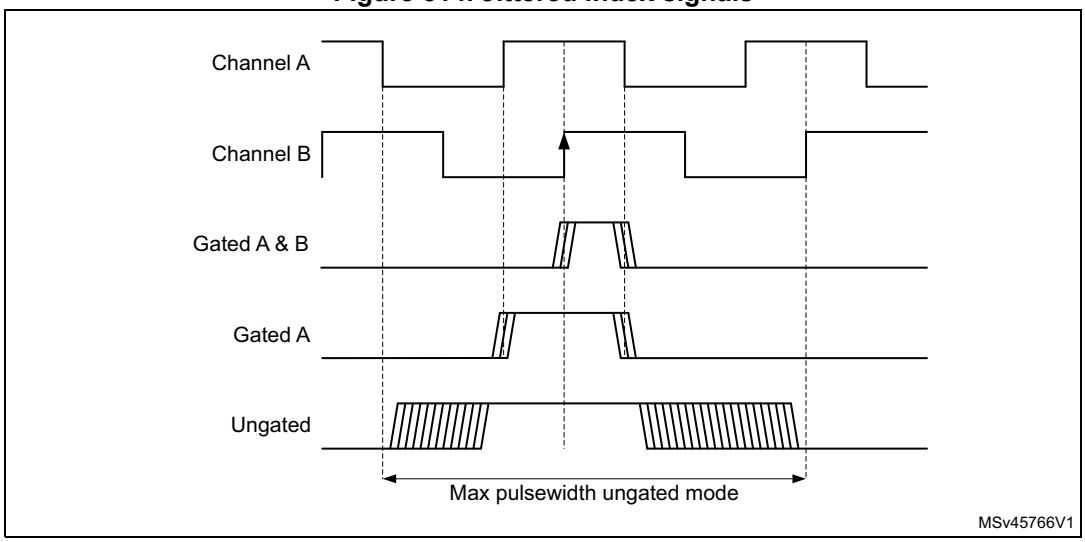
Figure 513. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 514*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 514. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 515* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 515. Index generation for IPOS[1:0] = 11

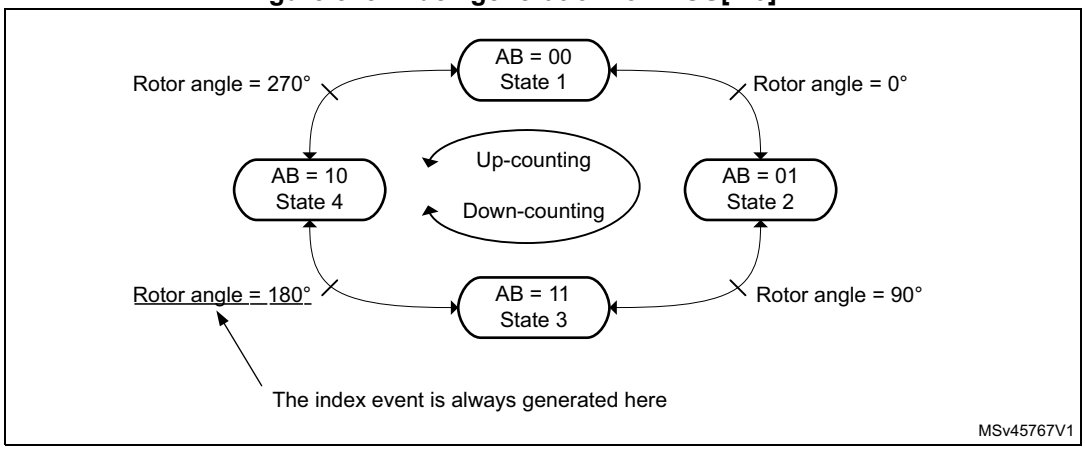


Figure 516 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 516. Counter reading with index gated on channel A (IPOS[1:0] = 11)

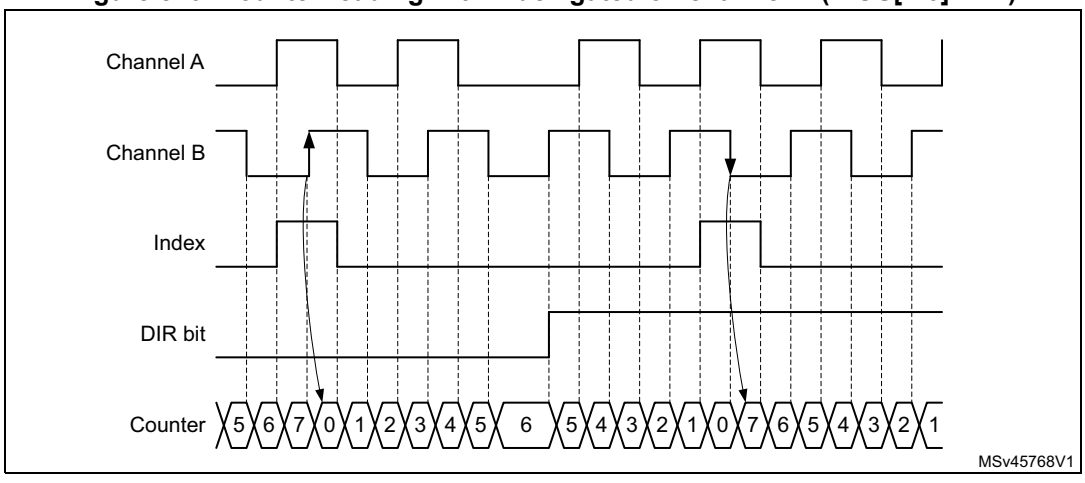


Figure 517. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 517. Counter reading with index ungated (IPOS[1:0] = 00)

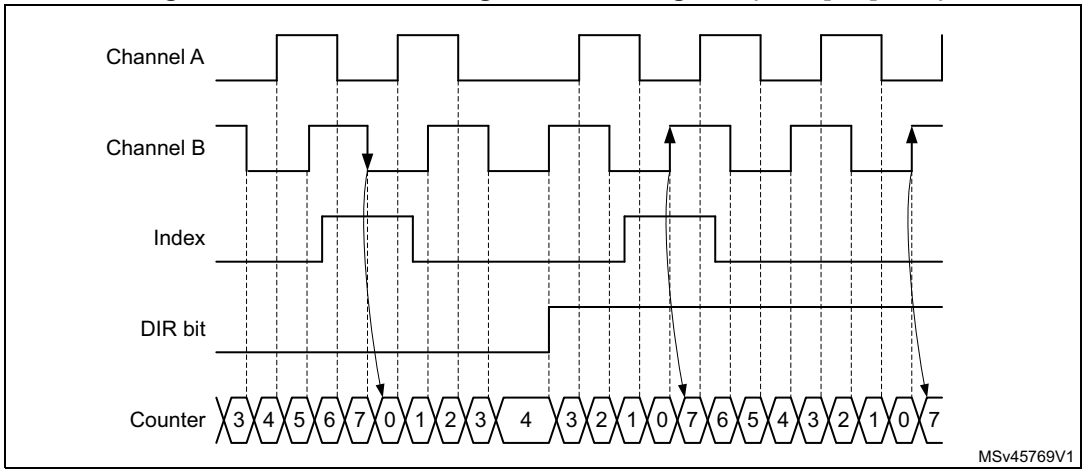


Figure 518. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 518. Counter reading with index gated on channel A and B

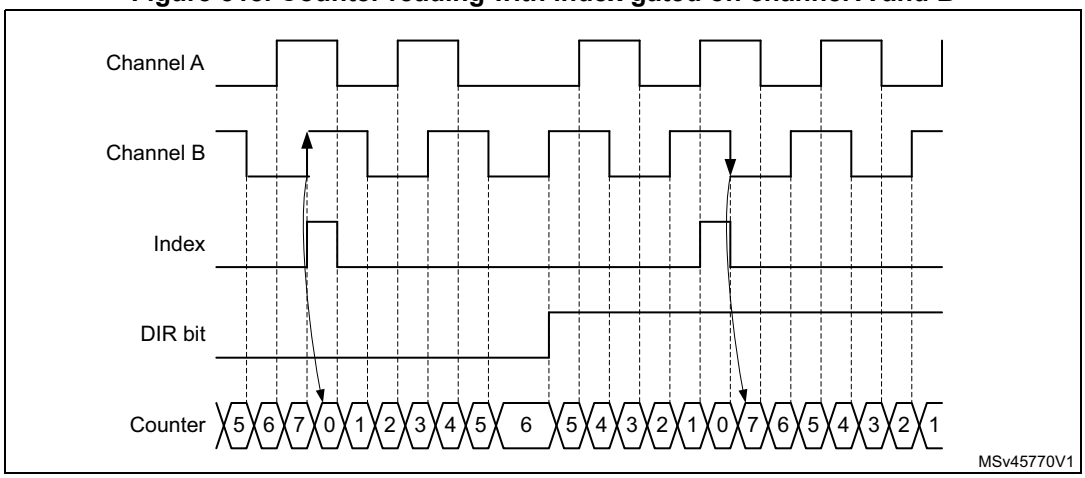


Figure 519 and *Figure 520* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 519. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

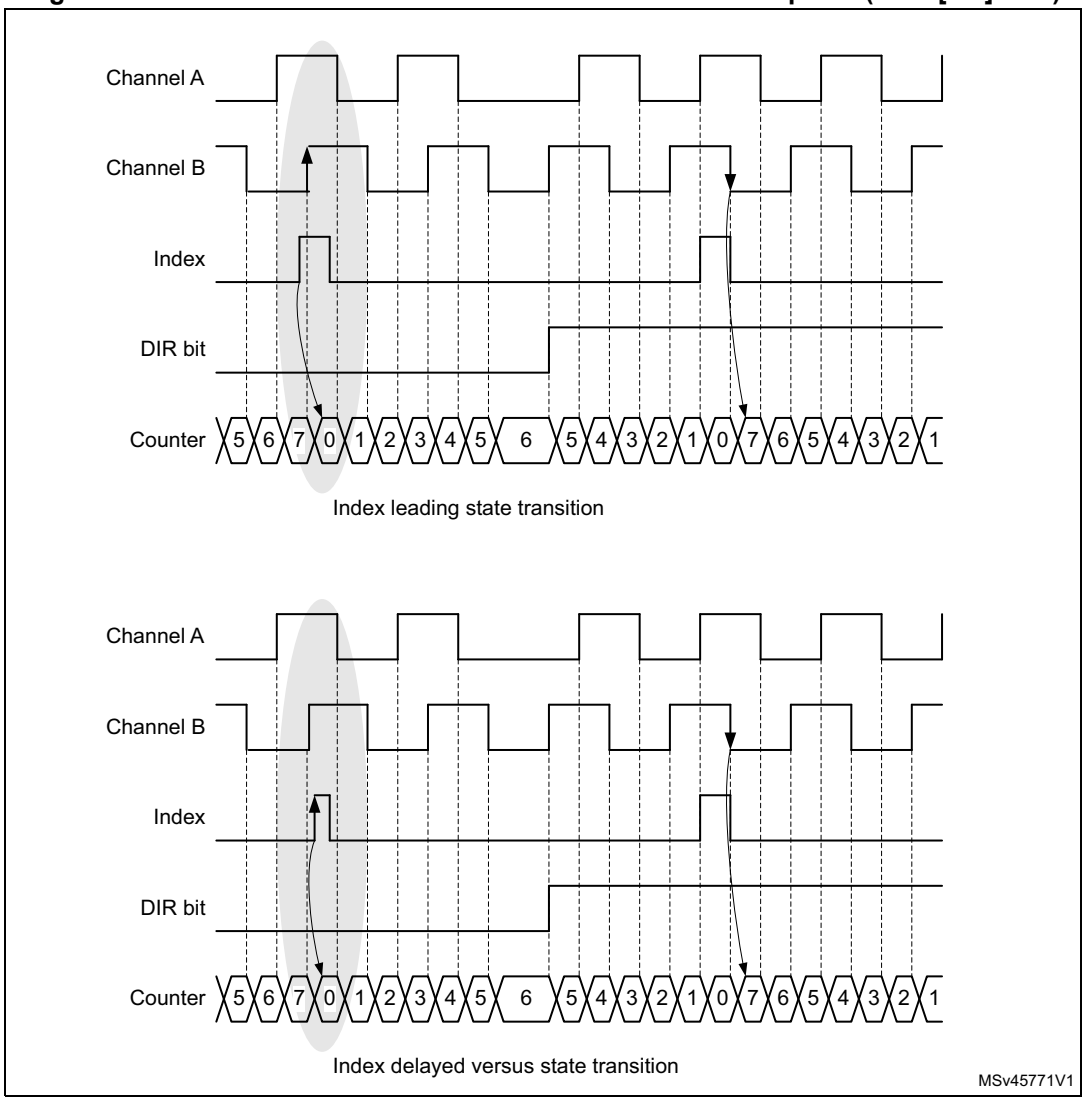


Figure 520. Counter reset Narrow index pulse (closer view, ARR = 0x07)

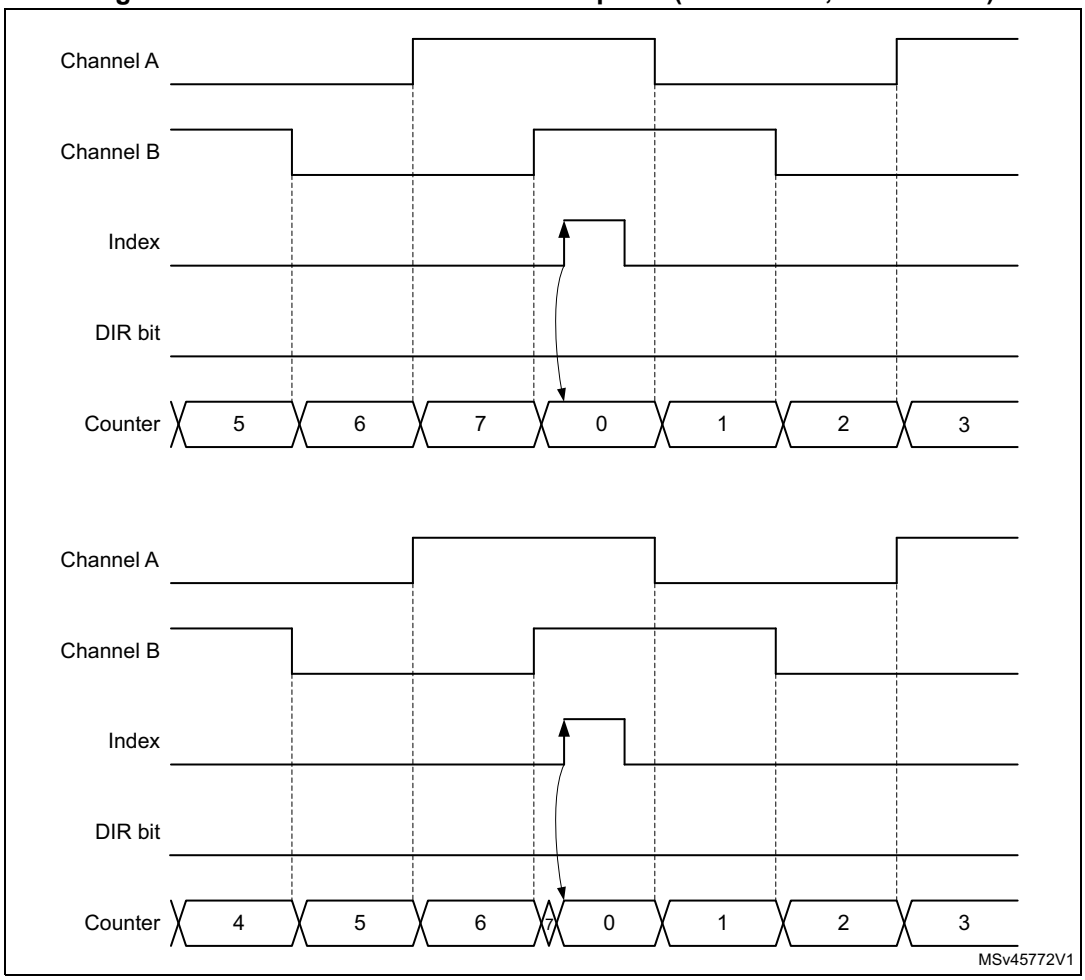
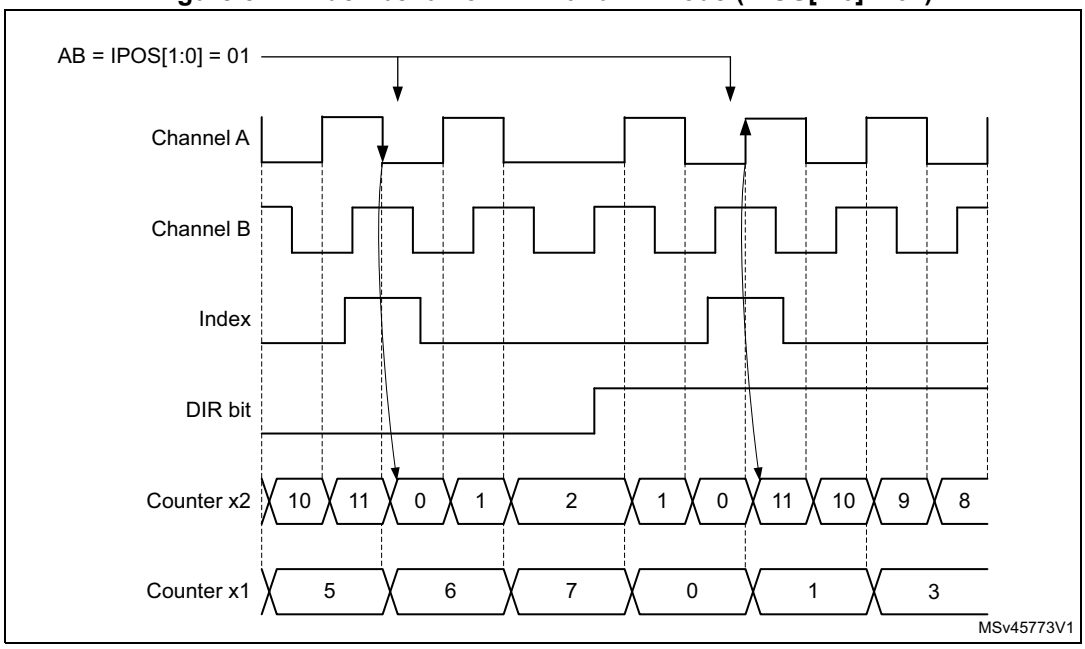


Figure 521 shows how the index is managed in x1 and x2 modes.

Figure 521. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

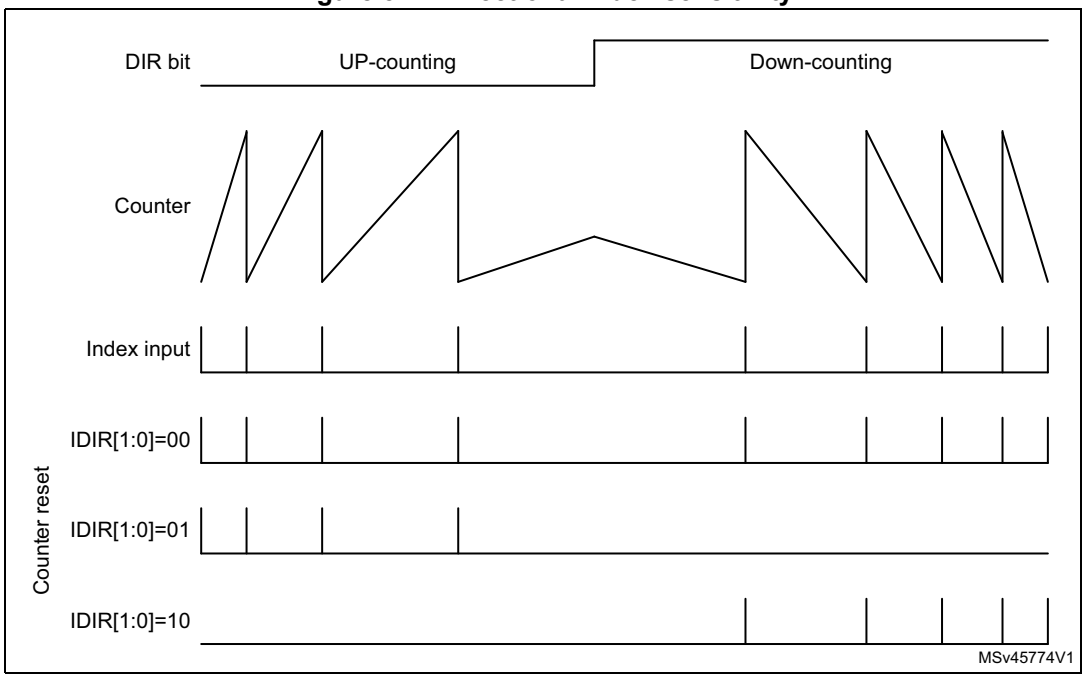


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 522 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

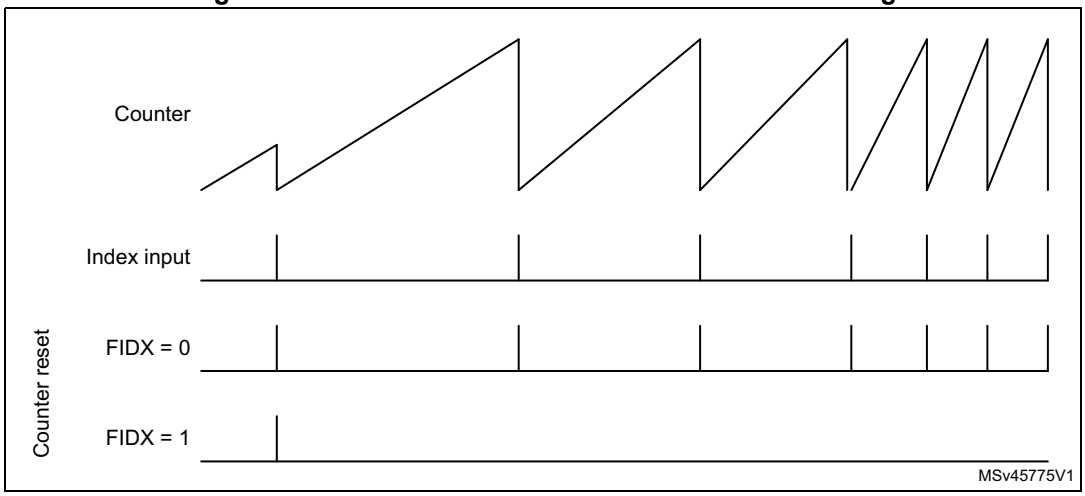
Figure 522. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 523*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 523. Counter reset as function of FIDX bit setting



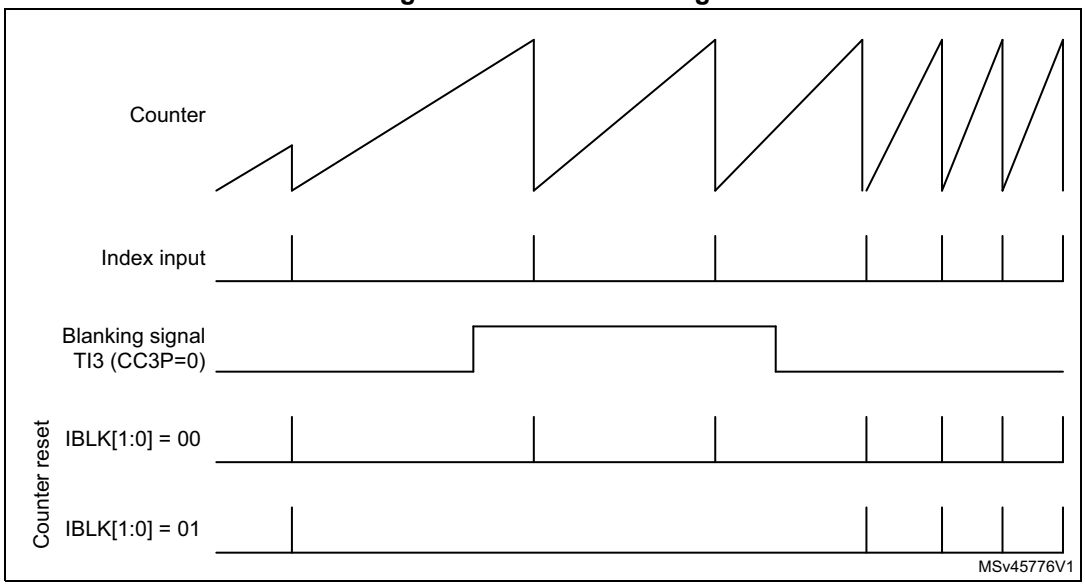
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 524* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 524. Index blanking



Index management in nonquadrature mode

Figure 525 and *Figure 526* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 525. Index behavior in clock + direction mode, IPOS[0] = 1

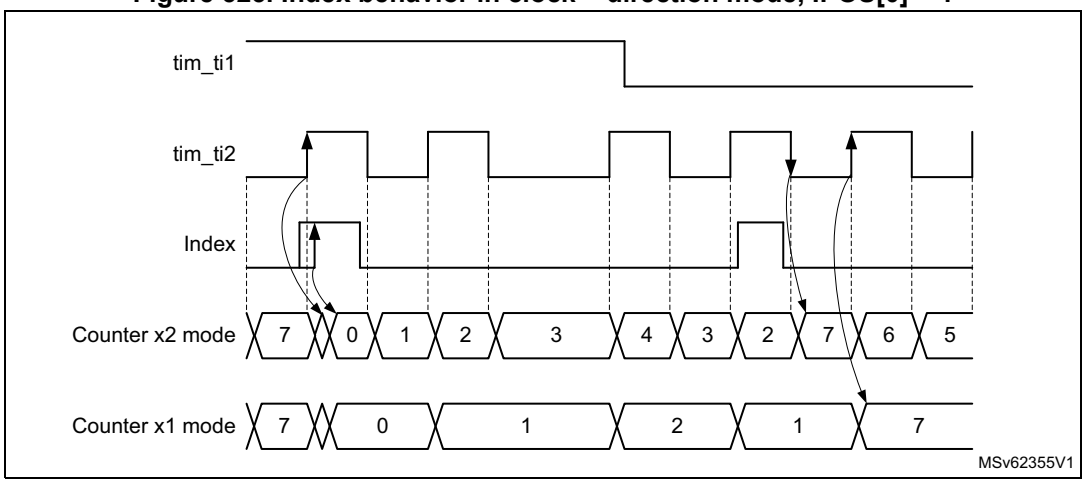
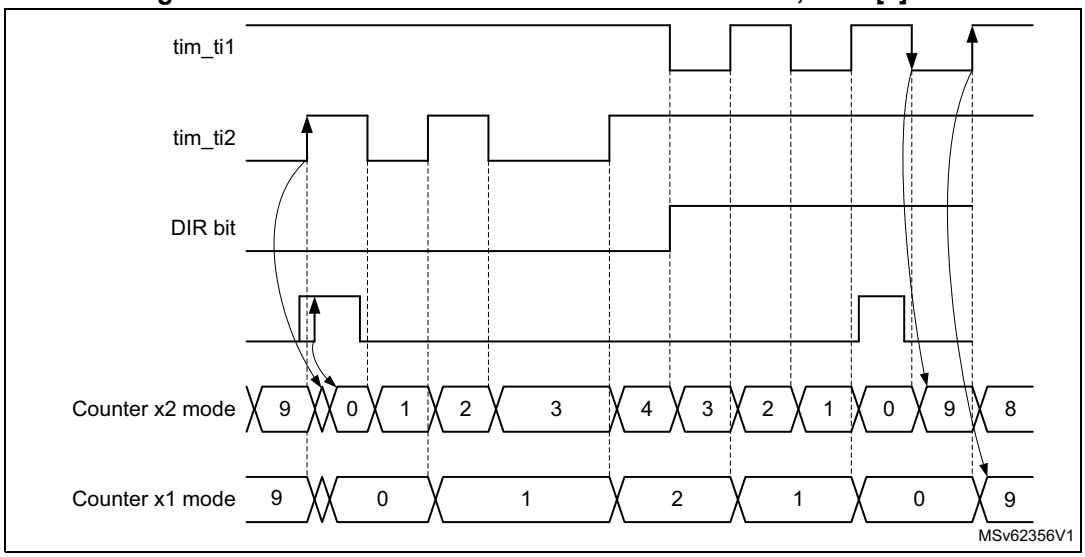


Figure 526. Index behavior in directional clock mode, IPOS[0] = 1

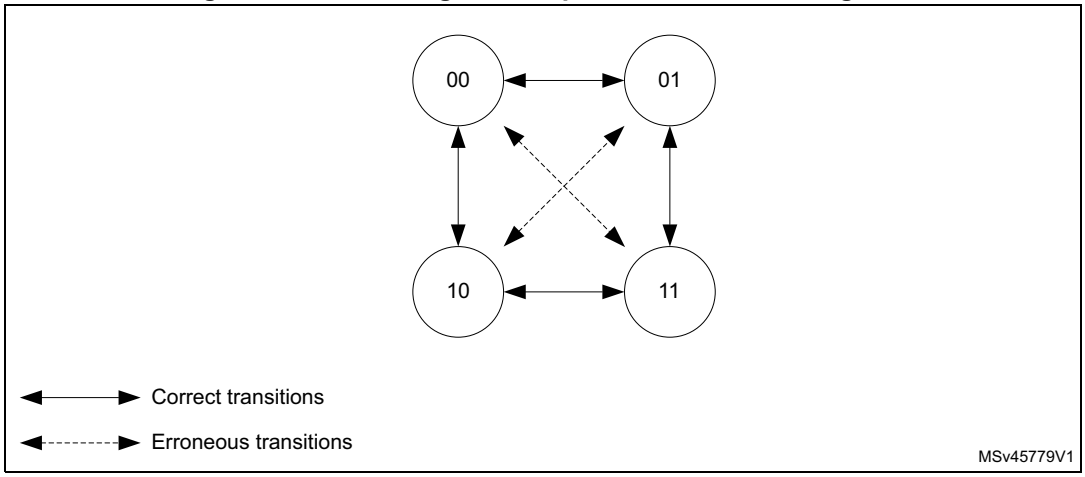


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 527.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 527. State diagram for quadrature encoded signals



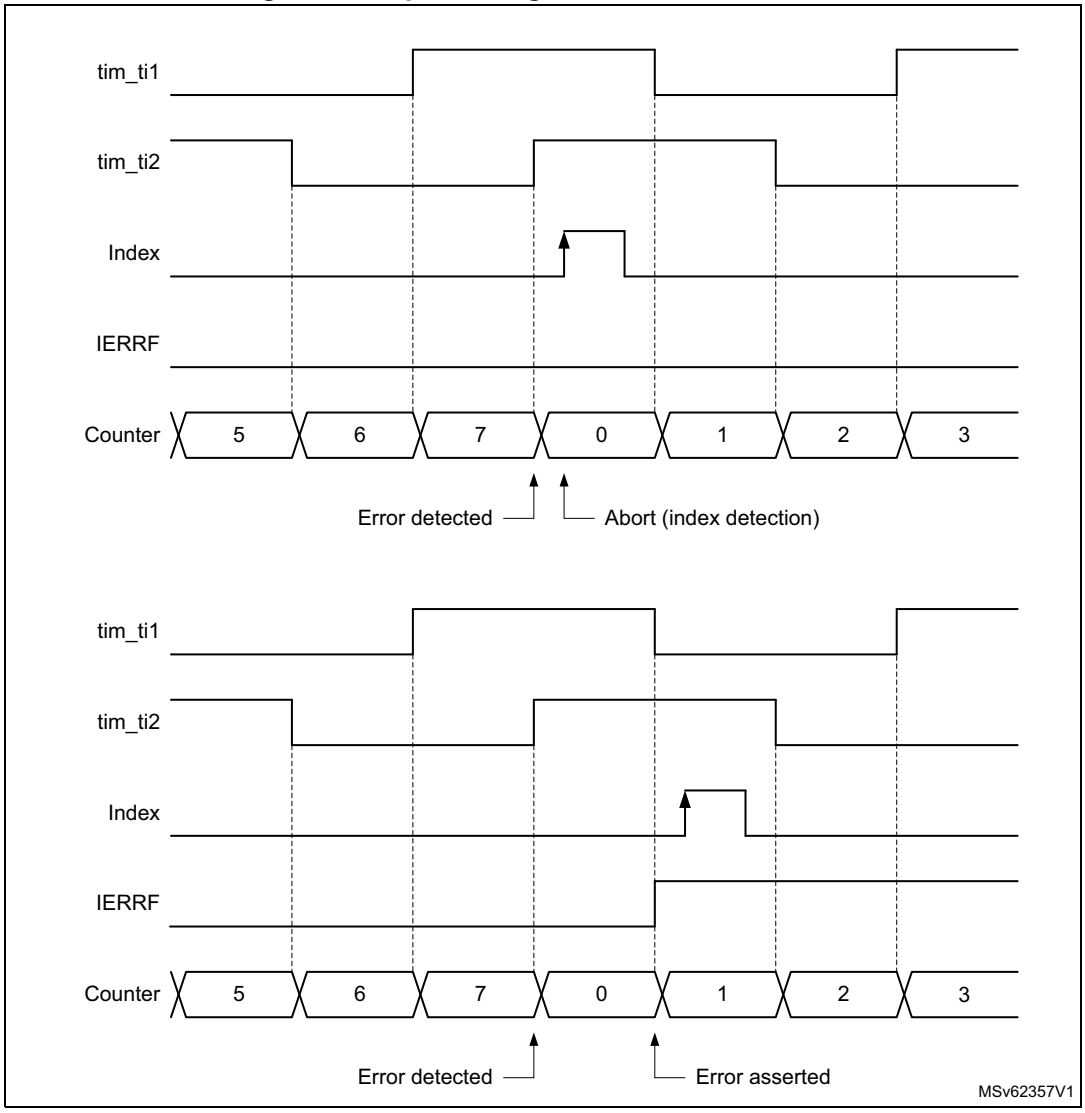
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

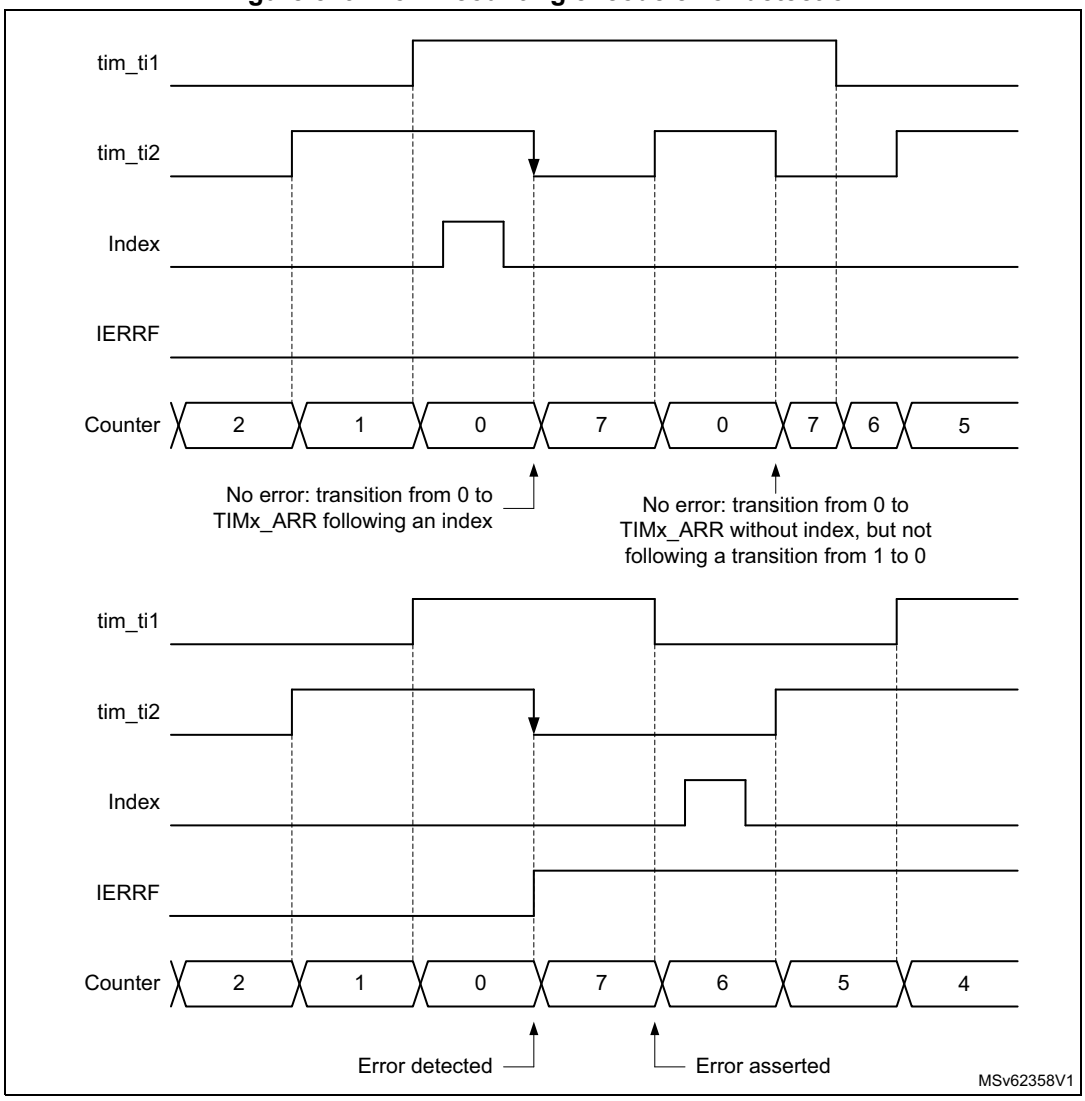
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 528*.

Figure 528. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 529*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 529. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

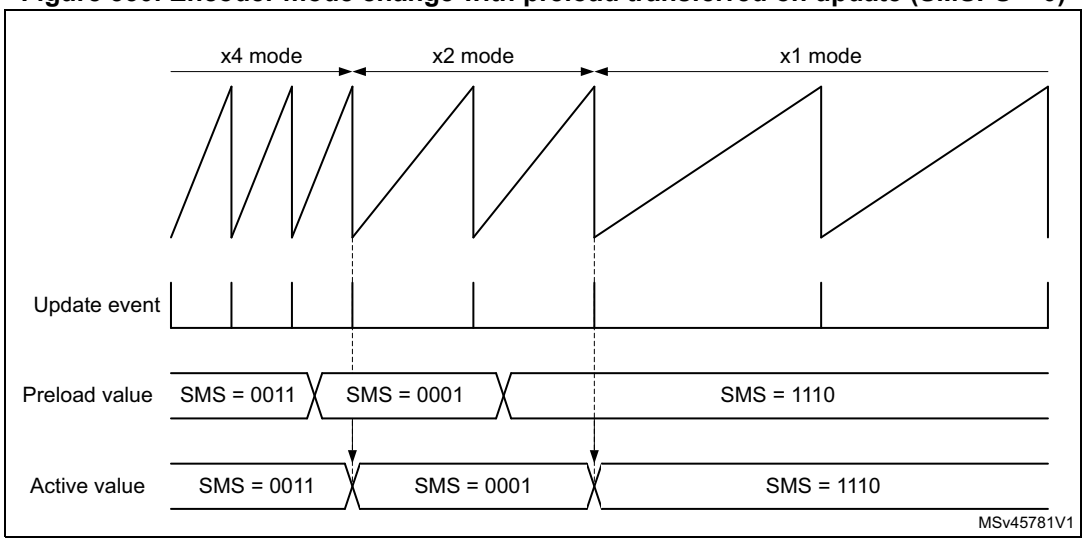
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 530*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 530. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

53.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

53.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

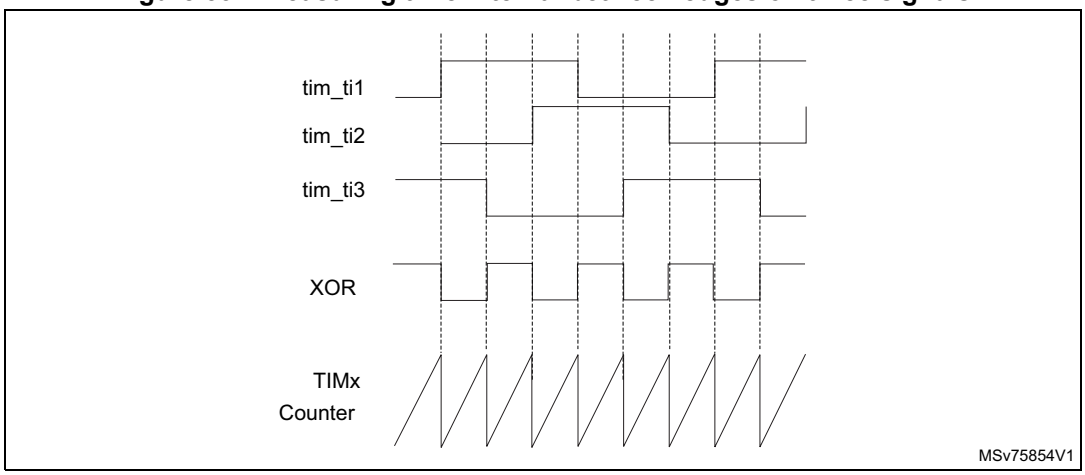
There is no latency between the UIF and UIFCPY flags assertion.

53.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 531*.

Figure 531. Measuring time interval between edges on three signals



53.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 532*. The "interfacing timer" captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 475*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

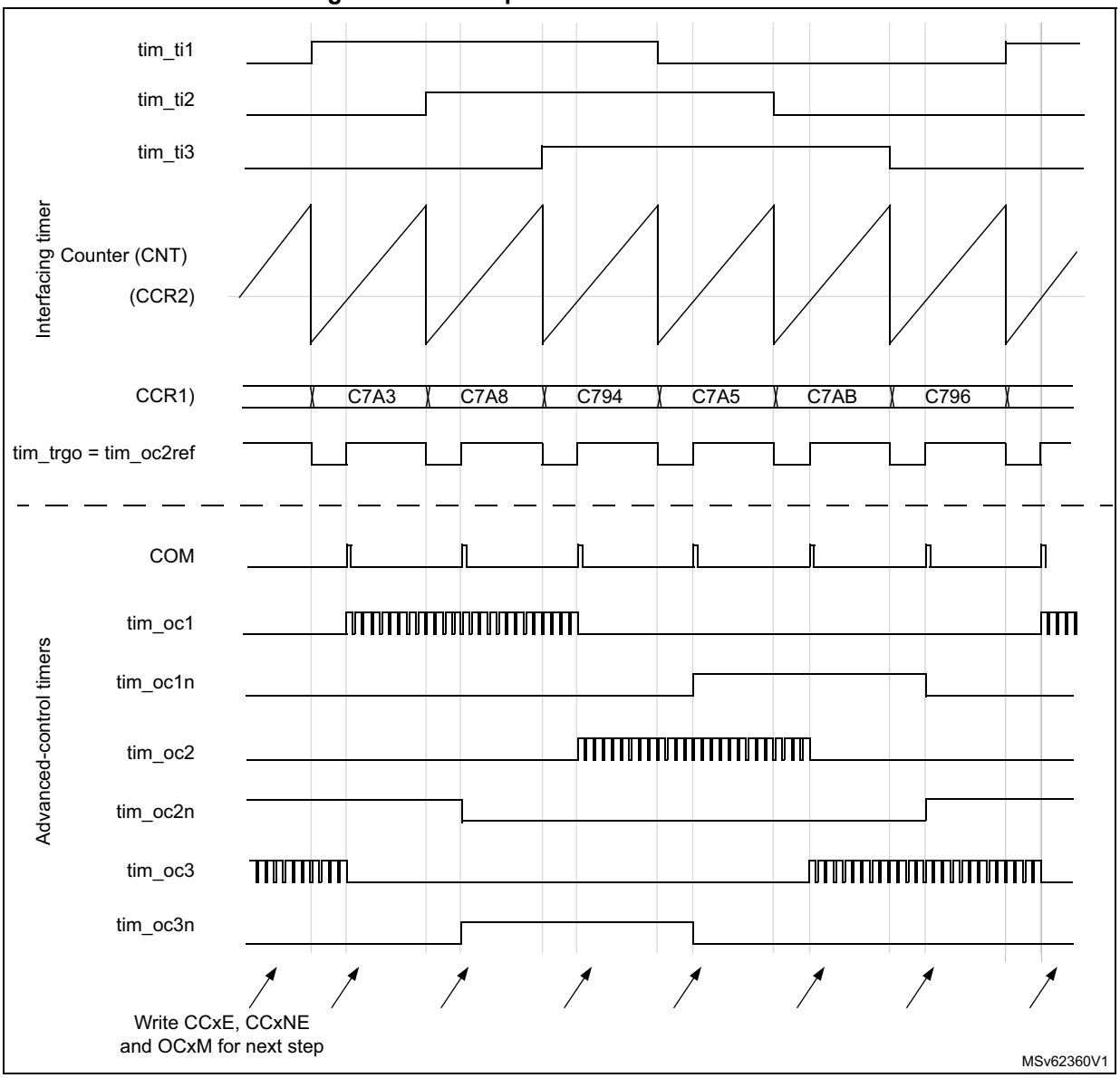
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 532 describes this example.

Figure 532. Example of Hall sensor interface



53.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

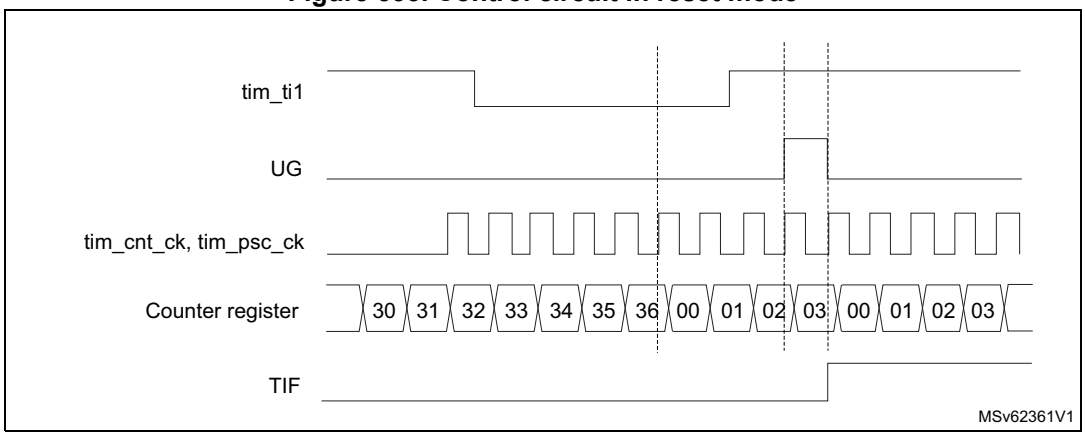
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 533. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

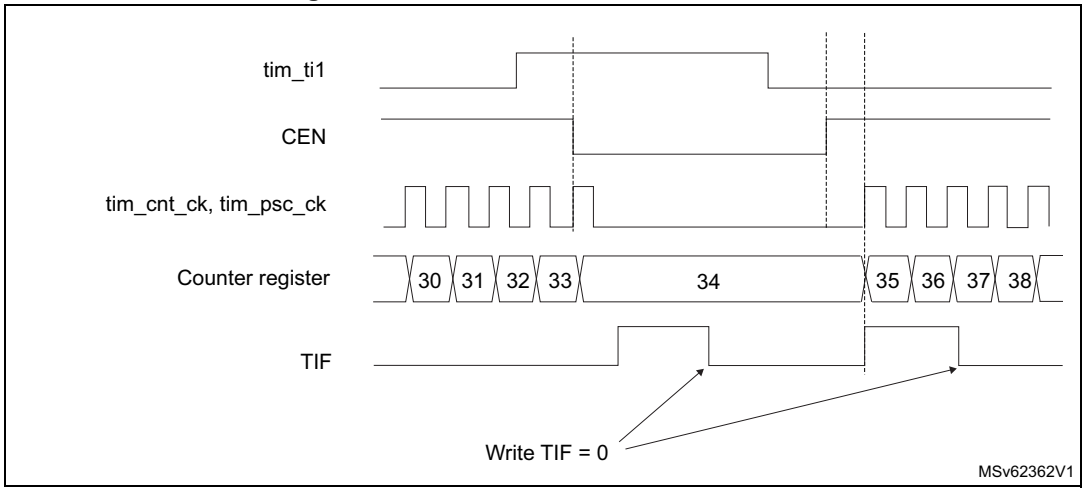
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 534. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

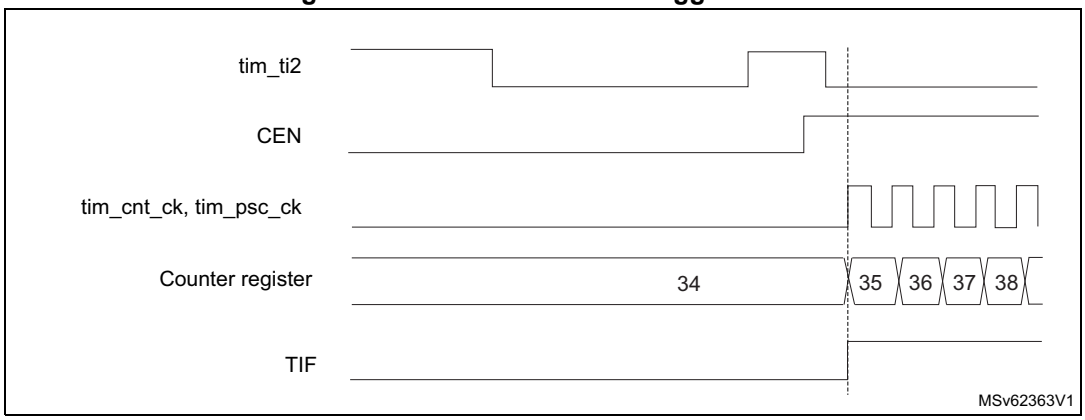
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 535. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

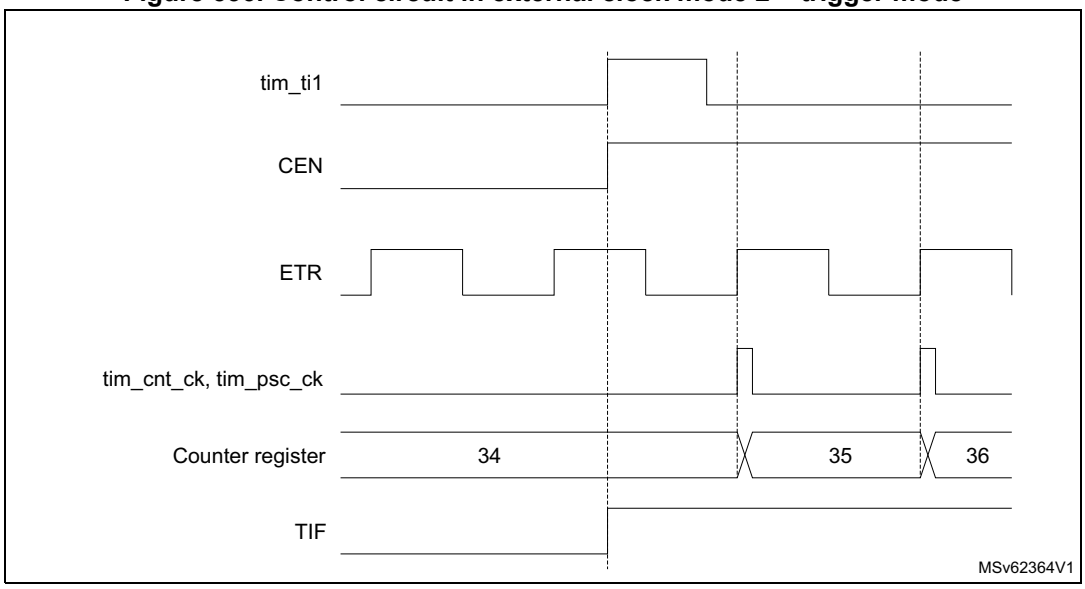
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 536. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

53.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 490*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

53.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 53.6.29: TIMx DMA control register (TIMx_DCR)(x = 1, 8)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address.
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (see note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

53.3.33 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

Table 535. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

53.3.34 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

53.4 TIM1/TIM8 low-power modes

Table 536. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

53.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 537*.

Table 537. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

53.6 TIM1/TIM8 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

53.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

53.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

53.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Table 524: Internal trigger connection* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

53.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

53.6.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 53.6.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

53.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

53.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

53.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

53.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

53.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

53.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 538* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 538. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0, tim_ocxn = 0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered).	
			1	0		
			1	1	Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

53.6.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

53.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

53.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 53.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

53.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

53.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

53.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

53.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

53.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

53.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 495: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 495: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

53.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

53.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

53.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

53.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

53.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input
00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

53.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

53.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 53.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

53.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 53.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

53.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

53.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

53.6.31 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 1, 8)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width
0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width
0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear
0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

53.6.32 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 1, 8)

Address offset: 0x3F0

Reset value: 0x2232 2146

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
0x0: Down-counter only
0x1: Up-counter only
0x2: Up/Down counter allowing center-aligned mode
Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
0x0: DMA requests are not possible
0x1: Single DMA requests possible
0x2: All DMA features implemented
Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
0x0: no break implemented
0x1: break1 implemented without filter (brk + brk_acth connected)
0x2: break1 implemented with filter (brk + brk_acth connected)
0x3: break 1 and 2 implemented (brk + brk_acth + brk2)
Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
0x0: External trigger circuit not implemented, ocrefx cleared by ocref_clr input
0x1: External trigger circuit implemented, ocrefx cleared by ocref_clr input
0x2: External trigger circuit implemented, ocrefx cleared by etrf signal
0x3: External trigger circuit implemented, ocrefx cleared by ocref_clr or etrf (1 control bit "OCCS" added in GPT_SMCR)
Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
0x0: repetition counter not implemented
0x1: 8-bit repetition counter implemented
0x2: 16-bit repetition counter implemented
Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

53.6.33 TIMx IP version register (TIMx_VERR)(x = 1, 8)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

53.6.34 TIMx IP identification register (TIMx_IPIDR)(x = 1, 8)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

53.6.35 TIMx size ID register (TIMx_SIDR)(x = 1, 8)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

53.6.36 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 539. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value							0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 539. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIMx_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	TIMx_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIMx_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIMx_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		

Table 539. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL[3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	TIMx_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIMx_HWCFGR1	CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]				RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
	Reset value	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0
0x3F4	TIMx_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIMx_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIMx_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

54 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

54.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 54.4.23: Timer synchronization*.

54.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.

54.3 TIM2/TIM3/TIM4/TIM5 implementation

Table 540. STM32MP21xx general purpose timers(1)

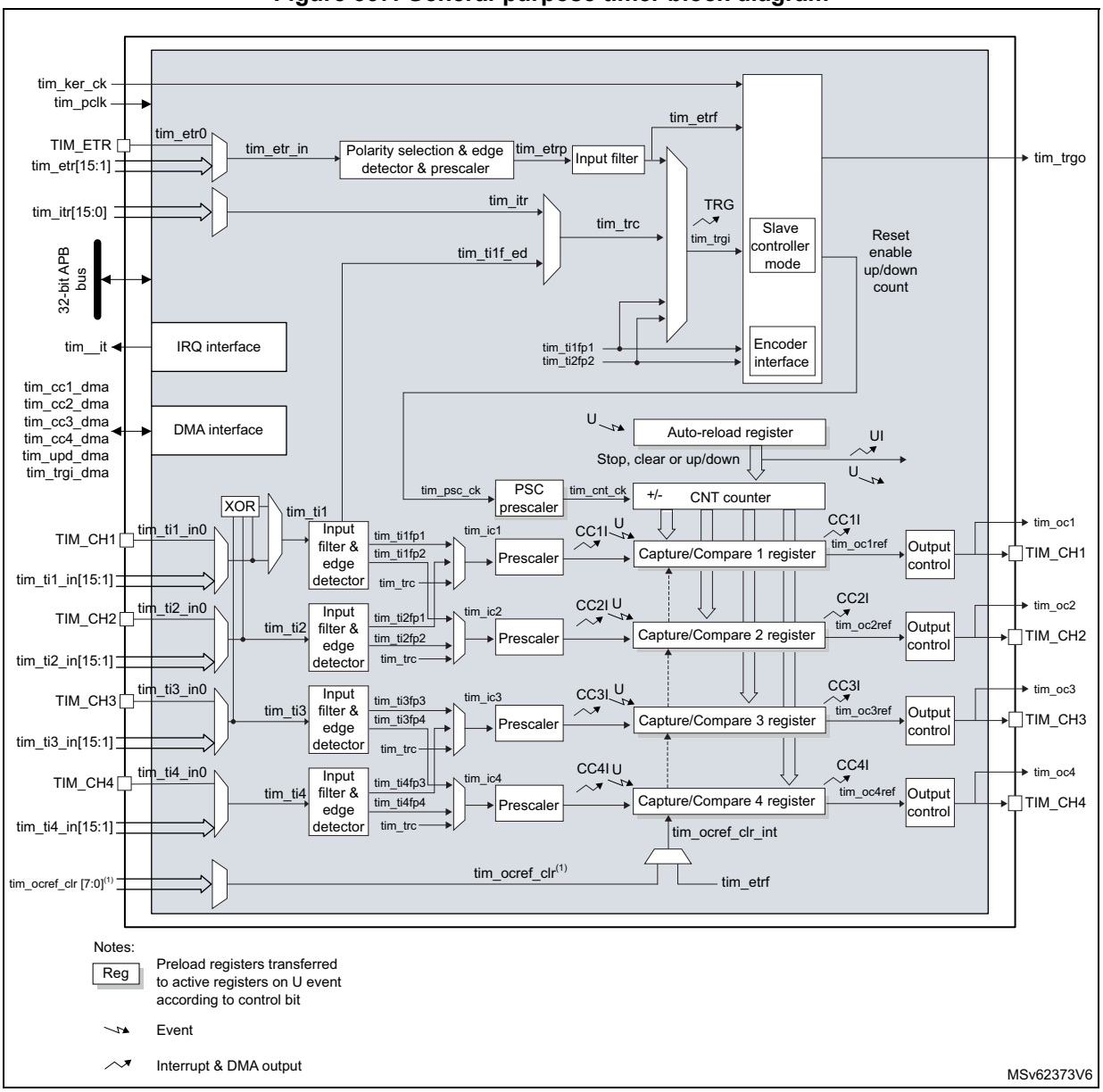
Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	32-bit	32-bit	32-bit
OCREF clear selection	No	No	No	No
Sources	tim_etrf	tim_etrf	tim_etrf	tim_etrf

1. Note: ‘X’ = supported, ‘-’= not supported.

54.4 TIM2/TIM3/TIM4/TIM5 functional description

54.4.1 Block diagram

Figure 537. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation*.

54.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 541 and *Table 542* in this section summarize the TIM inputs and outputs.

Table 541. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 542. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 542. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 543. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	eth1_pps	eth1_pps	Reserved	fdcan_tmp
tim_ti1_in2	Reserved	Reserved		fdcan_rtp
tim_ti1_in[15:3]	Reserved			

Table 544. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	eth2_pps	eth2_pps	Reserved	Reserved
tim_ti2_in[15:2]	Reserved			

Table 545. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in1	ptm_local_clock[17]	ptm_local_clock[17]	Reserved	Reserved
tim_ti3_in[15:2]	Reserved			

Table 546. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	ptm_local_clock[20]	ptm_local_clock[20]	Reserved	
tim_ti4_in2	tim12_trgo	tim12_trgo		
tim_ti4_in[15:3]	Reserved			

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 547. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim12_trgo	tim12_trgo	tim12_trgo	tim12_trgo
tim_itr7	tim13_oc1	tim13_oc1	tim13_oc1	tim13_oc1
tim_itr8	tim14_oc1	tim14_oc1	tim14_oc1	tim14_oc1
tim_itr9	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr10	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr11	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr12	Reserved			
tim_itr13	tim10_oc1	tim10_oc1	tim10_oc1	tim10_oc1
tim_itr14	tim11_oc1	tim11_oc1	tim11_oc1	tim11_oc1
tim_itr15	Reserved			fdcan_soc

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 548. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	Reserved	TIM2_ETR	TIM2_ETR	TIM2_ETR
tim_etr2	TIM3_ETR	Reserved	TIM3_ETR	TIM3_ETR

Table 548. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr3	TIM4_ETR	TIM4_ETR	Reserved	TIM4_ETR
tim_etr4	TIM5_ETR	TIM5_ETR	TIM5_ETR	Reserved
tim_etr5	LSE	Reserved	Reserved	
tim_etr6	sai1_fs_a	eth1_pps		sai2_fs_a
tim_etr7	sai1_fs_b	eth2_pps		sai2_fs_b
tim_etr8	eth1_pps	Reserved		usbh_sof
tim_etr9	eth2_pps			otg_sof
tim_etr10	ptm_local_clock[0]	ptm_local_clock[0]		Reserved
tim_etr11	Reserved	Reserved		
tim_etr12				
tim_etr13				
tim_etr[15:14]				

54.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC

register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 538 and *Figure 539* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 538. Counter timing diagram with prescaler division change from 1 to 2

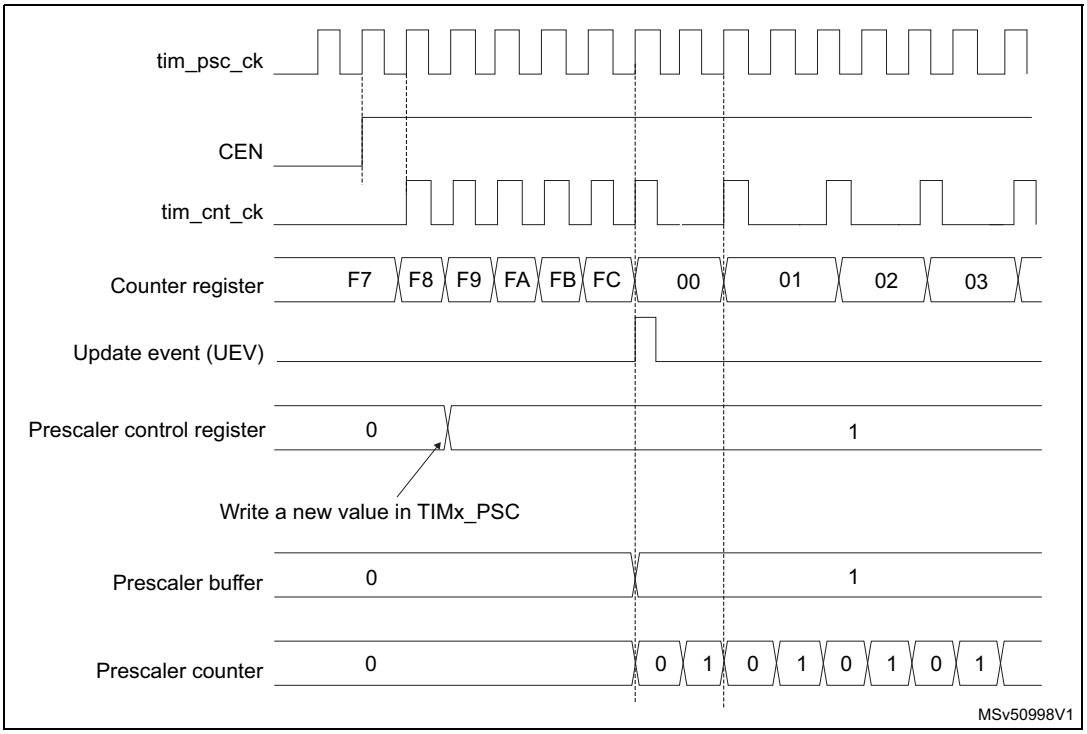
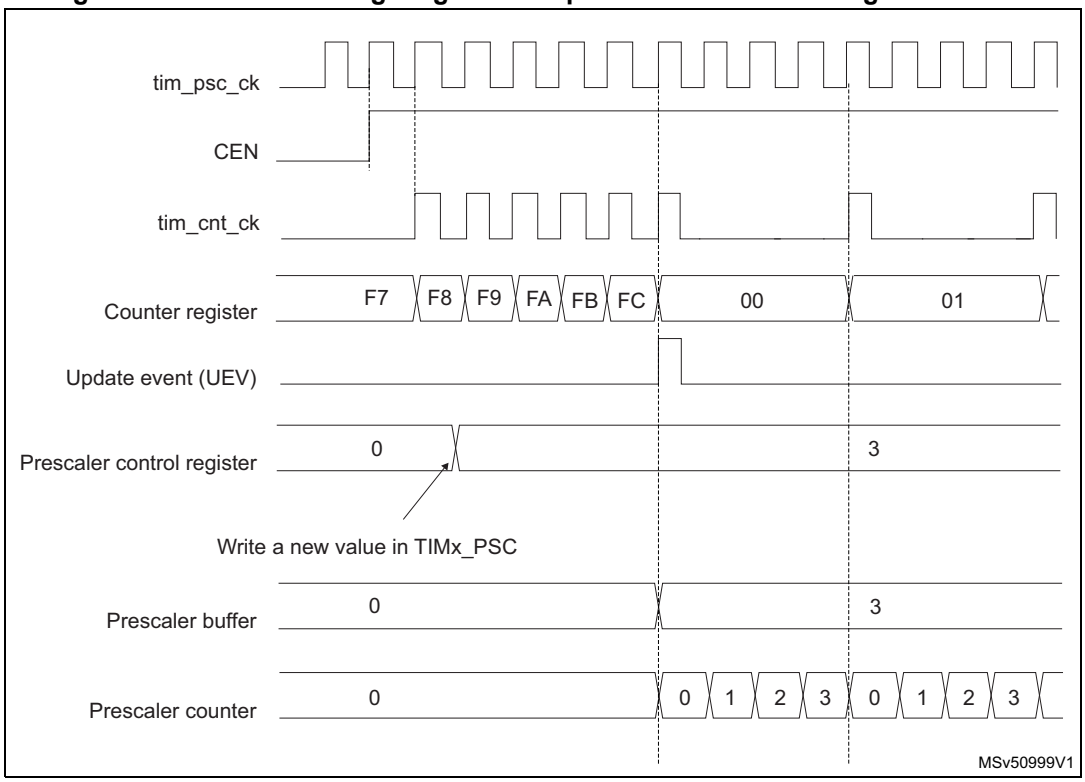


Figure 539. Counter timing diagram with prescaler division change from 1 to 4



54.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 540. Counter timing diagram, internal clock divided by 1

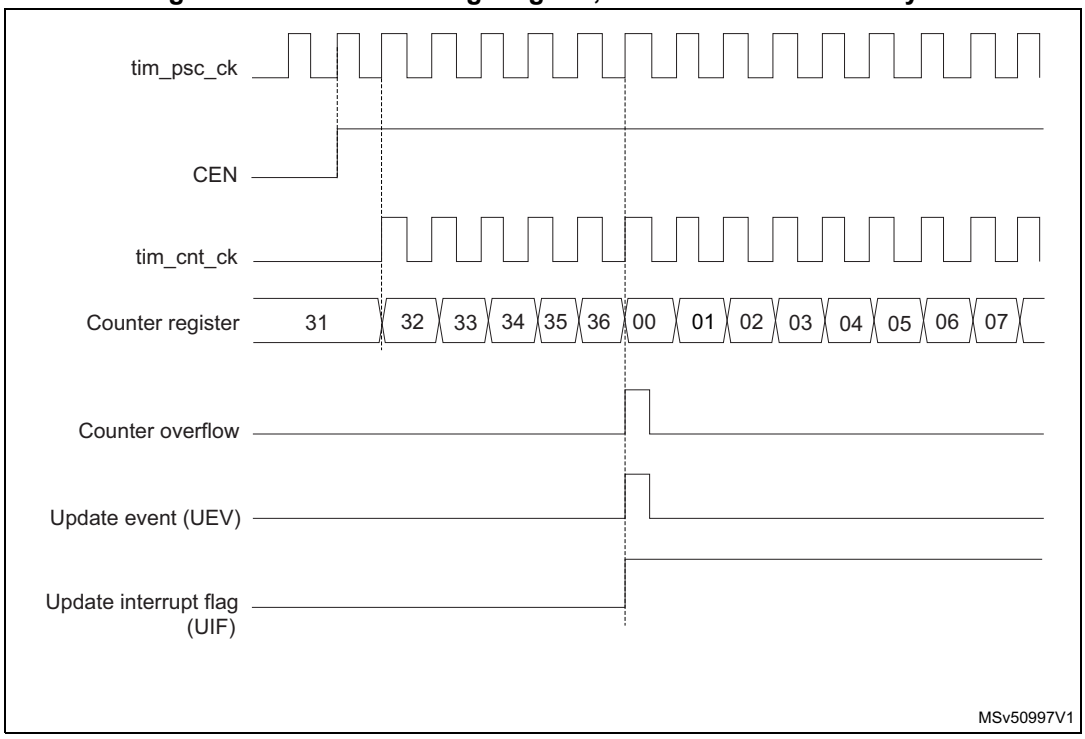


Figure 541. Counter timing diagram, internal clock divided by 2

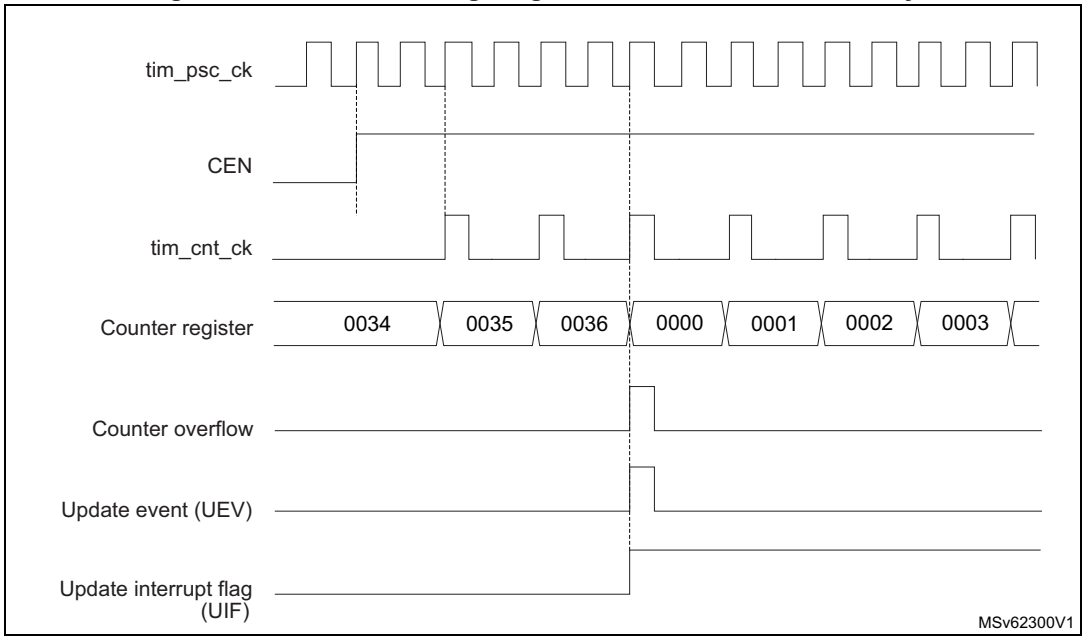


Figure 542. Counter timing diagram, internal clock divided by 4

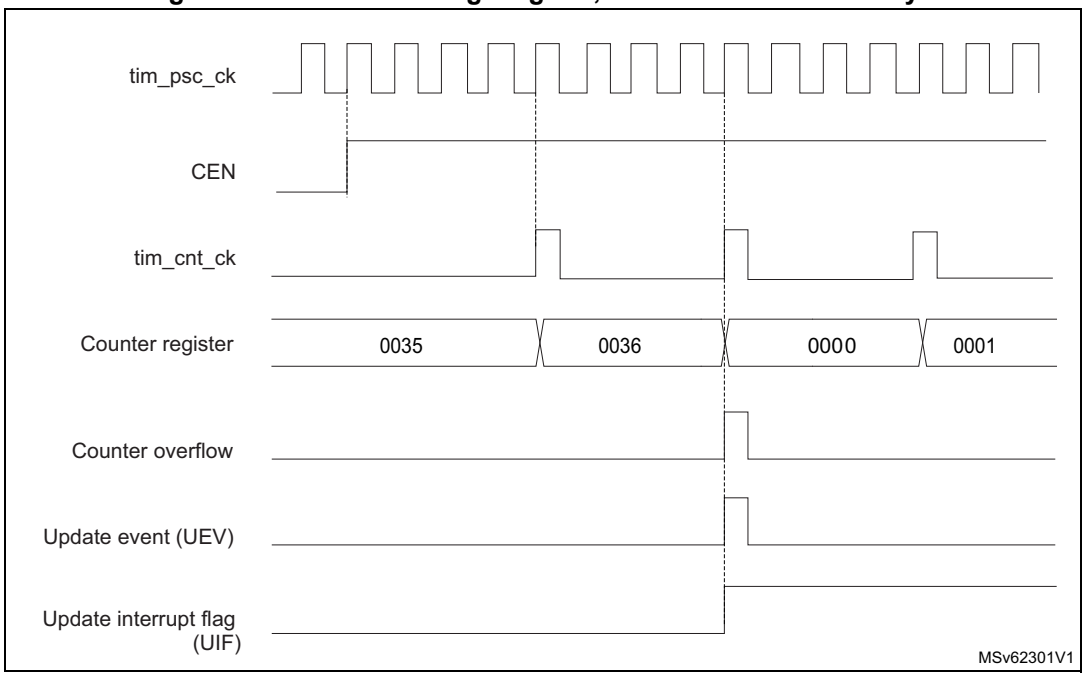


Figure 543. Counter timing diagram, internal clock divided by N

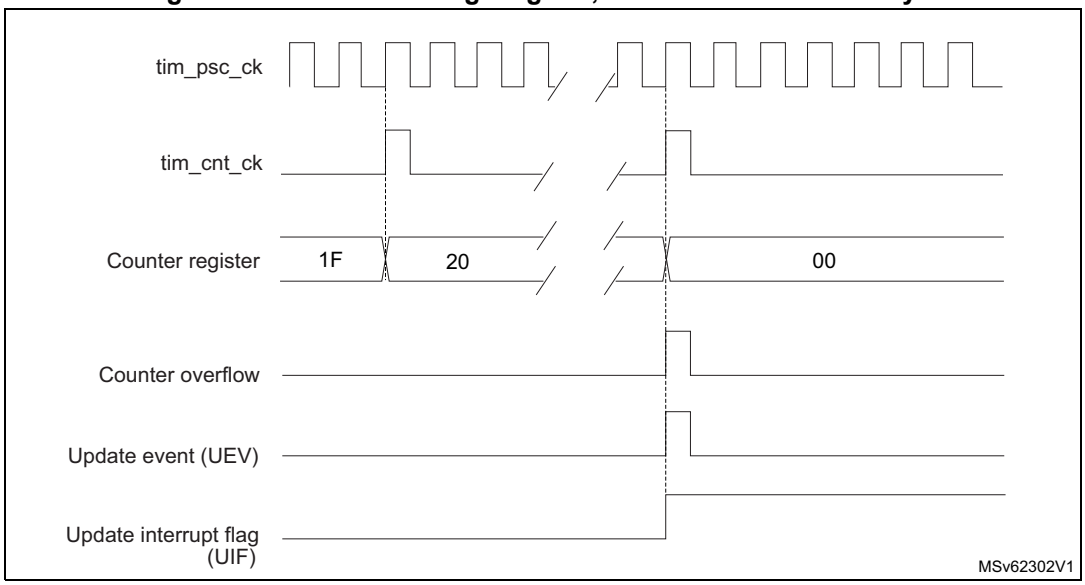


Figure 544. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

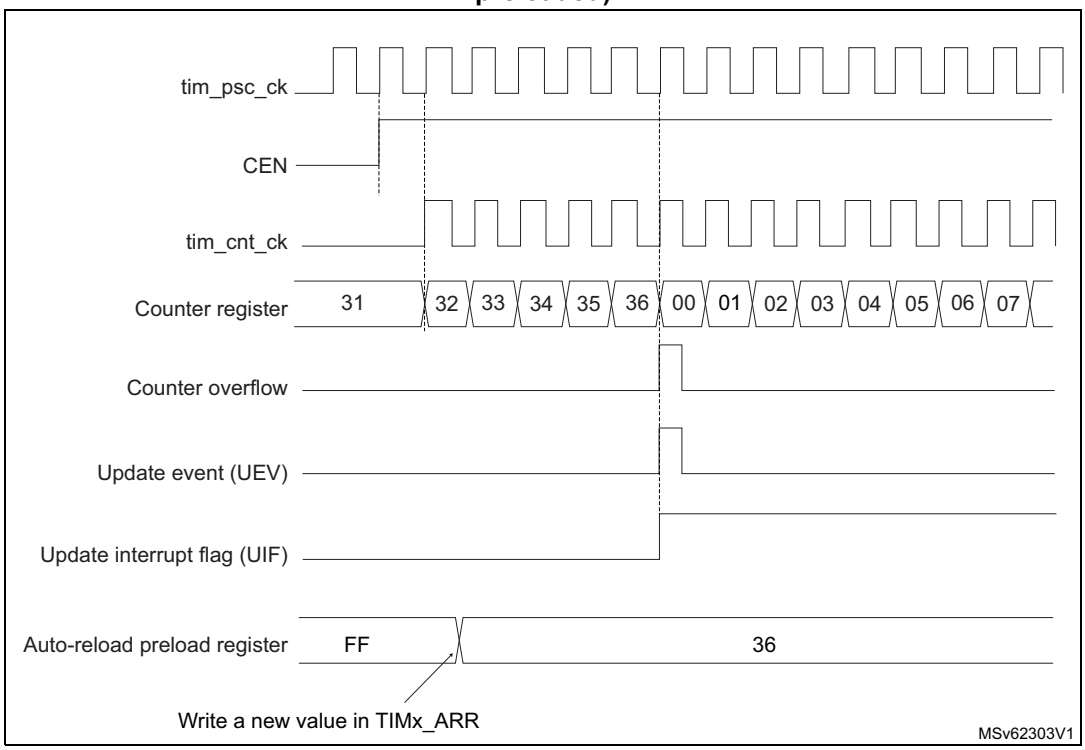
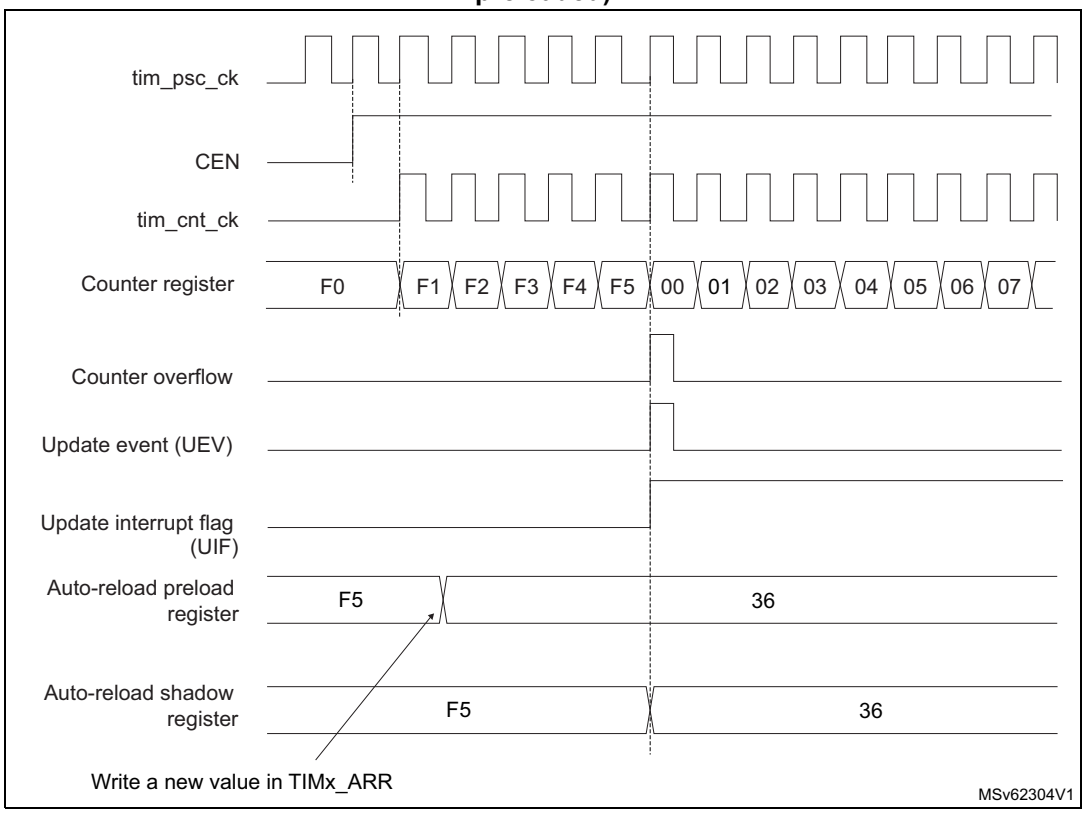


Figure 545. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 546. Counter timing diagram, internal clock divided by 1

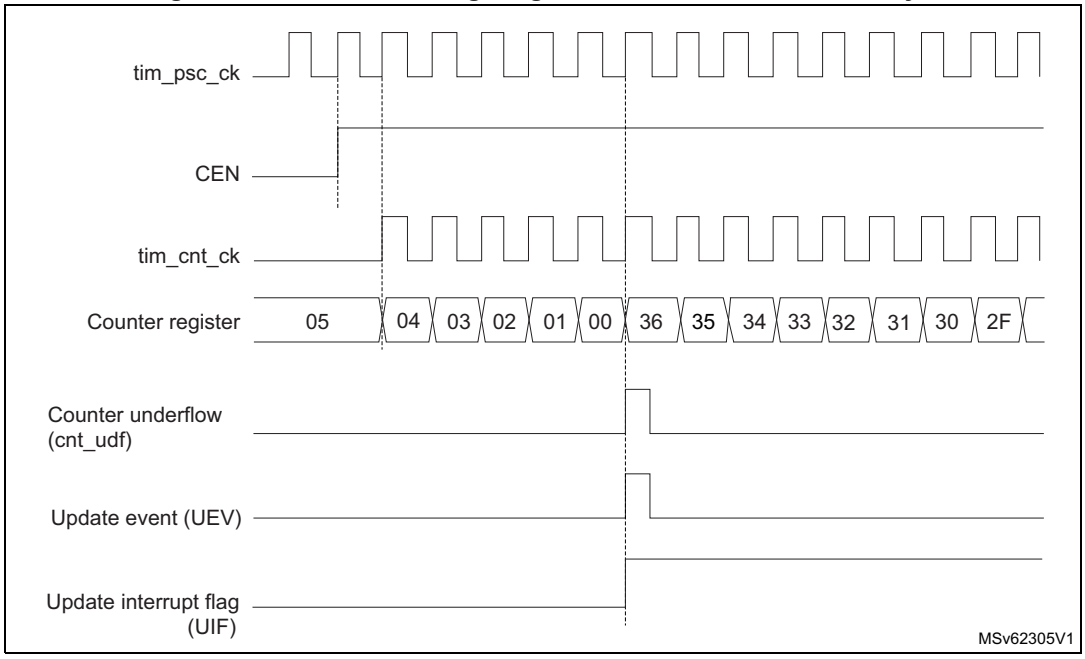


Figure 547. Counter timing diagram, internal clock divided by 2

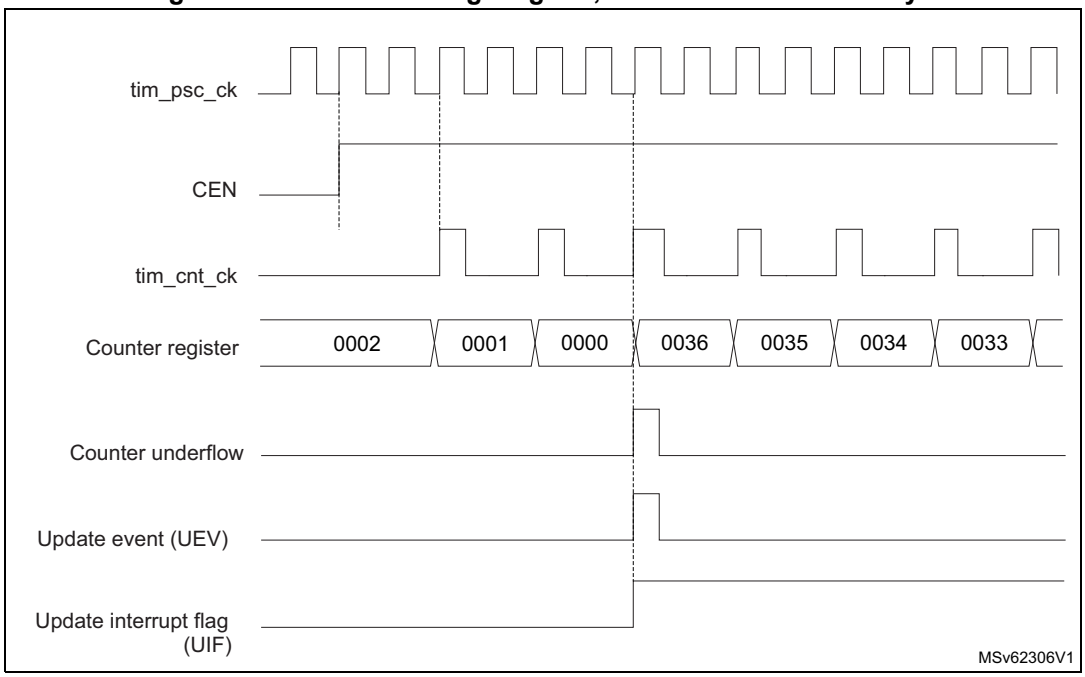


Figure 548. Counter timing diagram, internal clock divided by 4

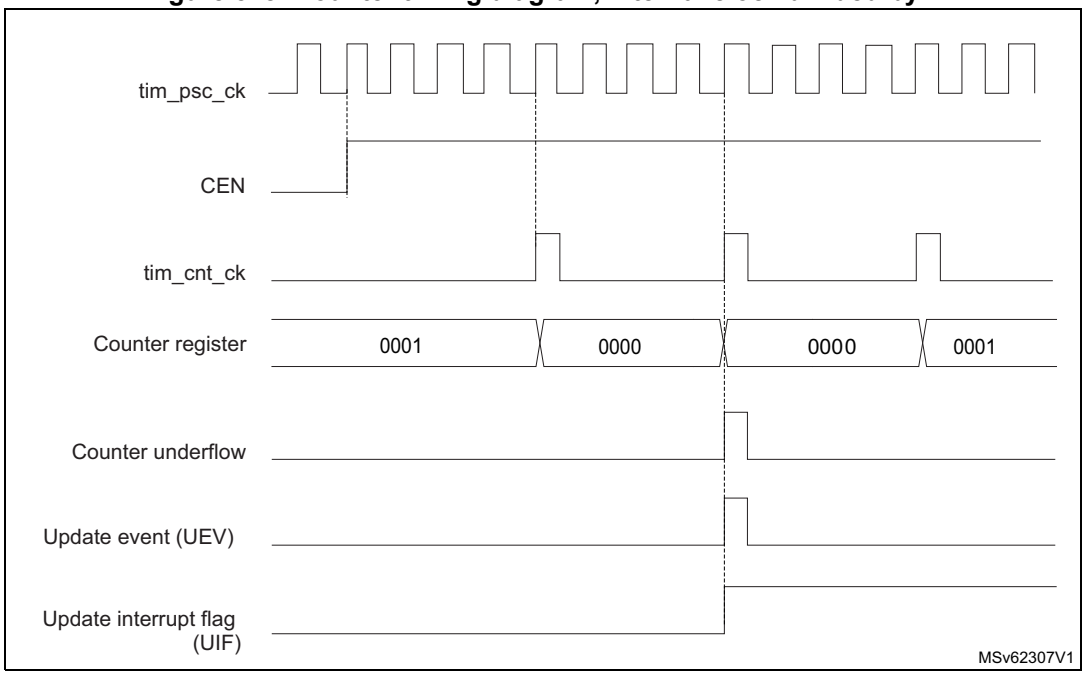


Figure 549. Counter timing diagram, internal clock divided by N

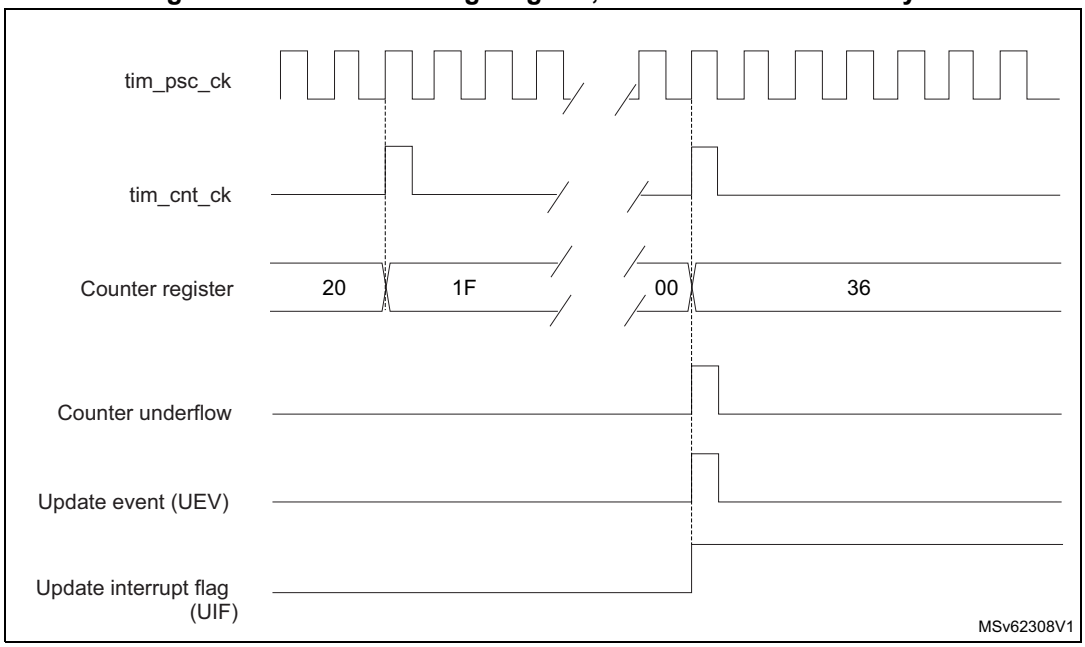
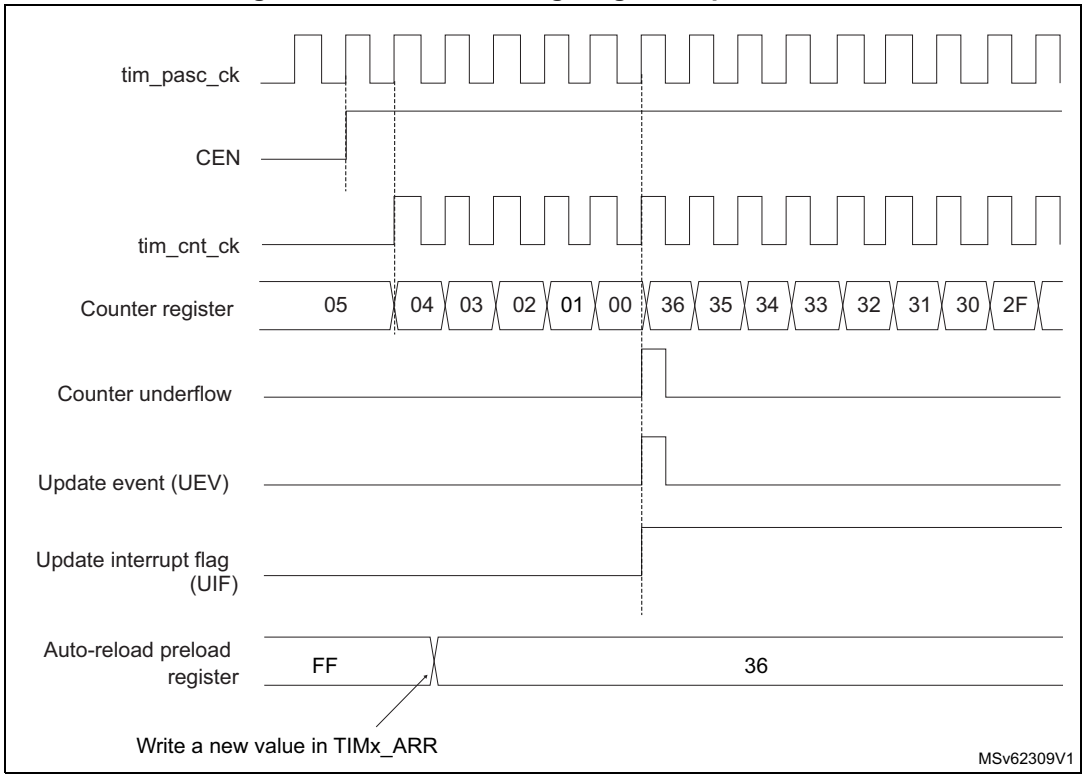


Figure 550. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

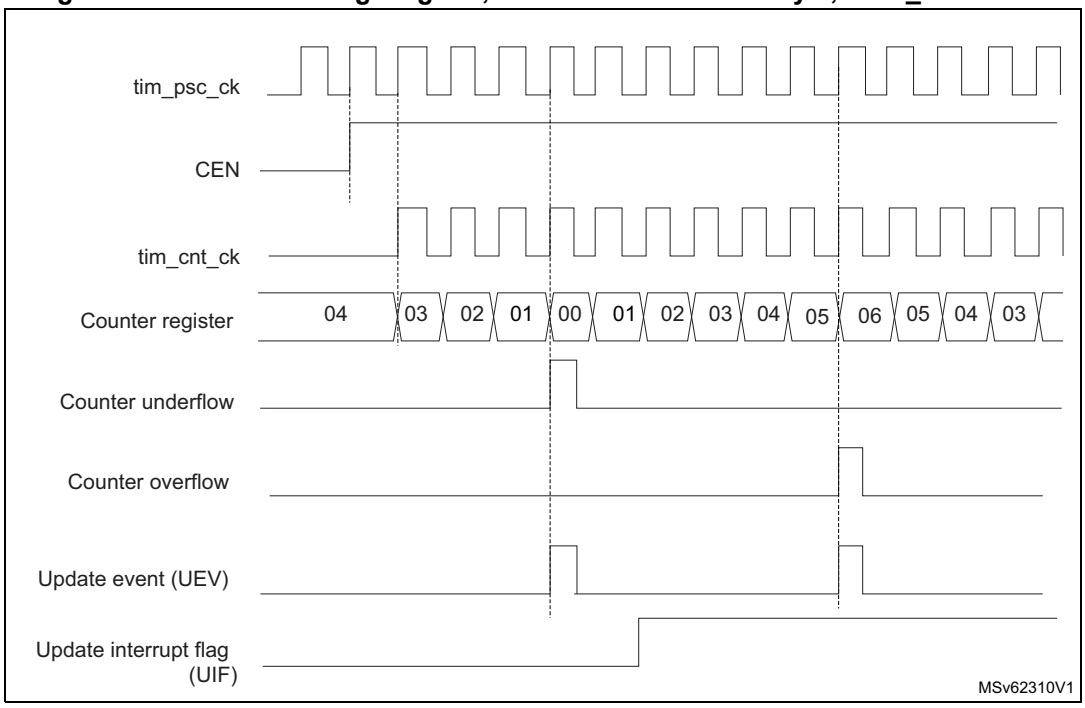
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 551. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 54.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)*).

Figure 552. Counter timing diagram, internal clock divided by 2

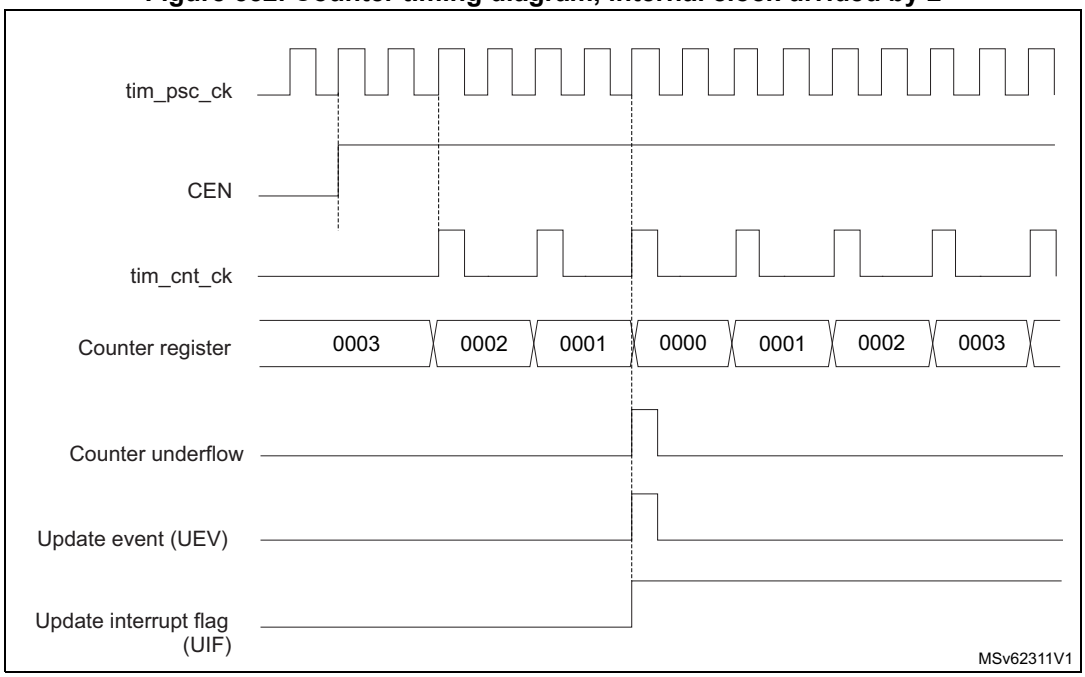
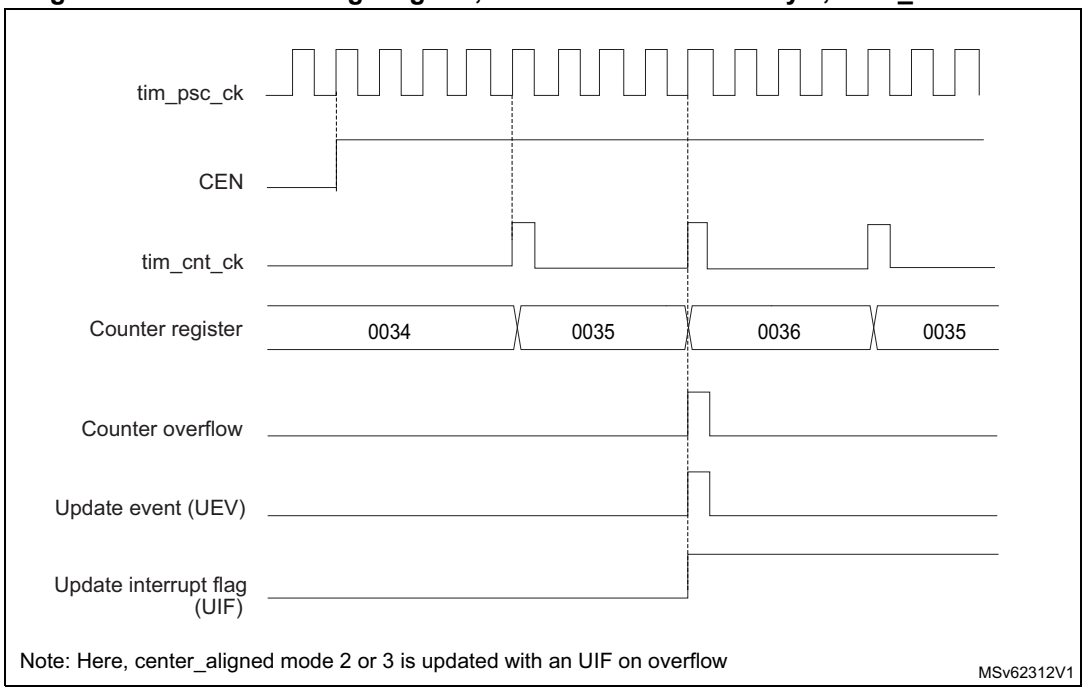


Figure 553. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 554. Counter timing diagram, internal clock divided by N

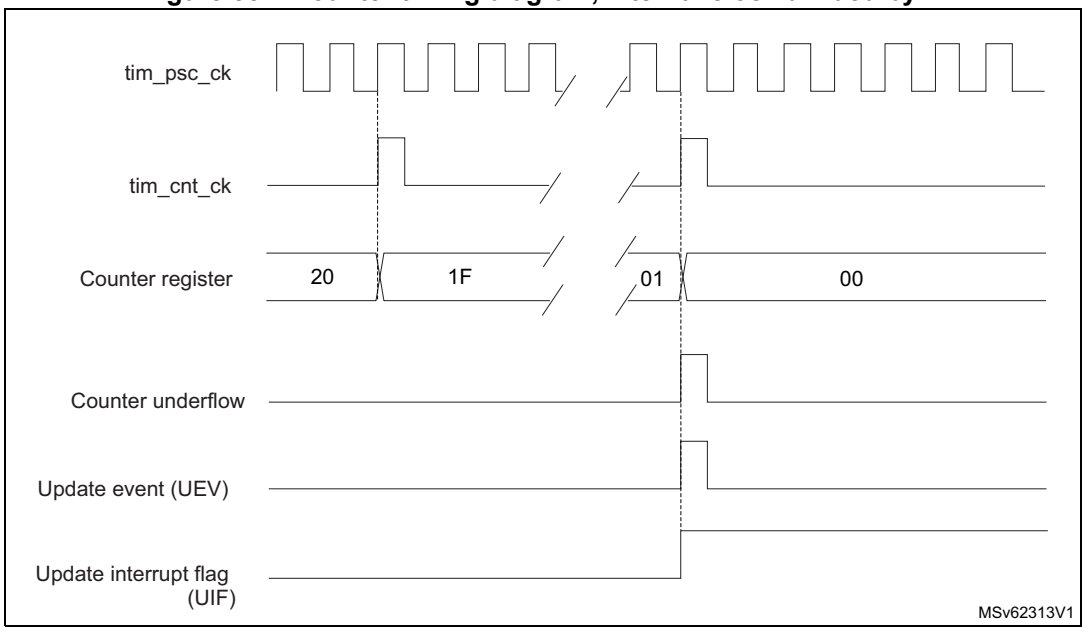


Figure 555. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

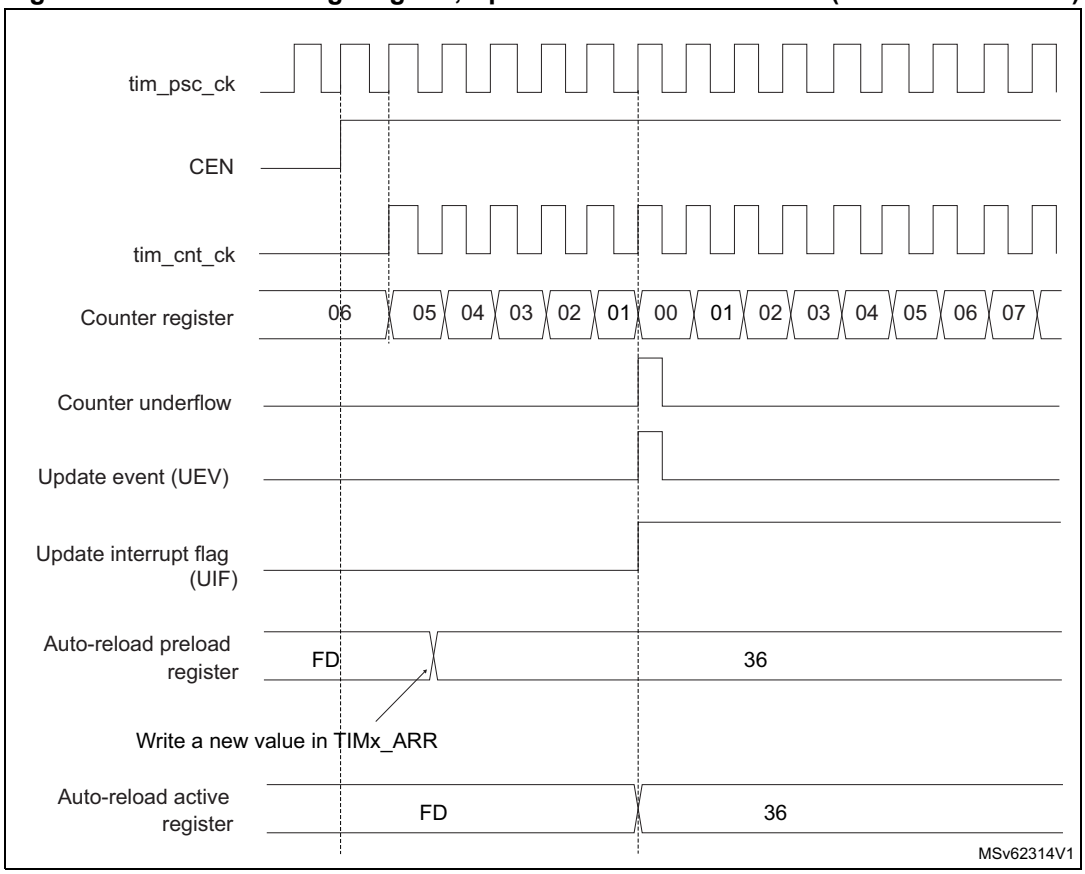
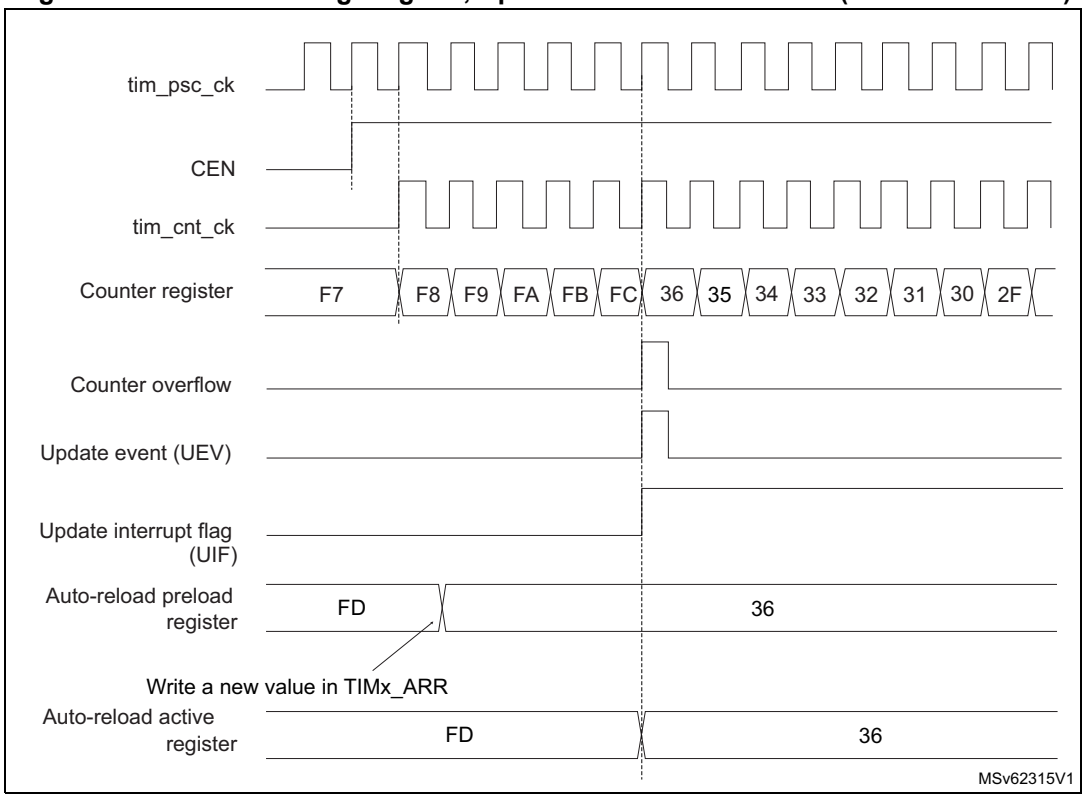


Figure 556. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



54.4.5 Clock selection

The counter clock can be provided by the following clock sources:

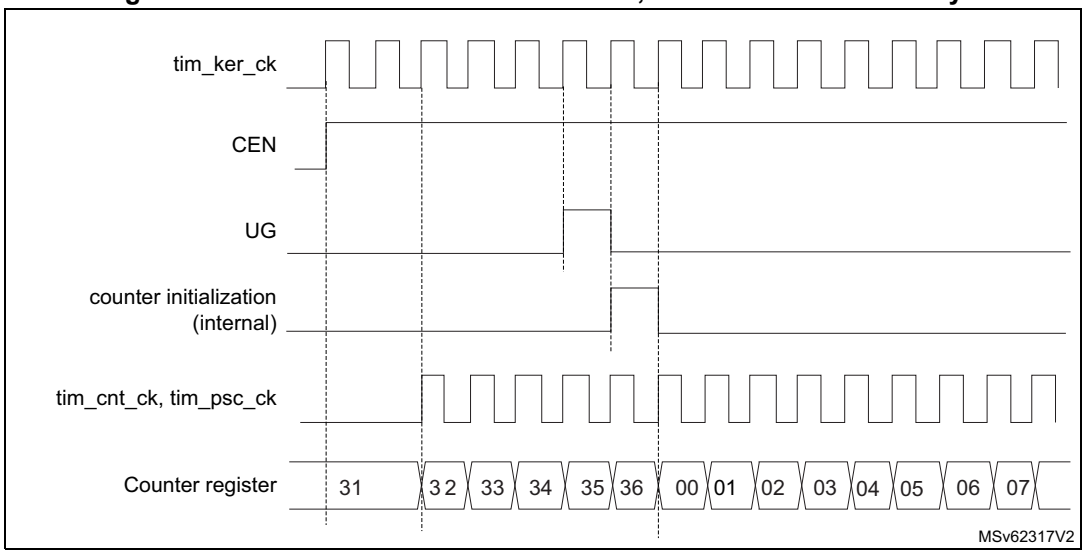
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 557 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

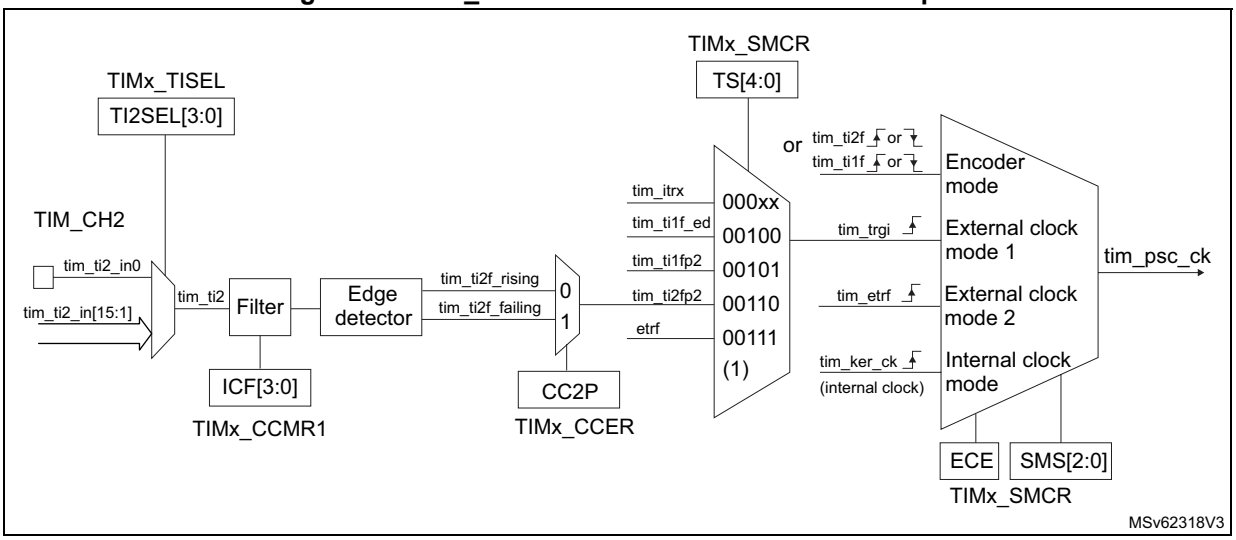
Figure 557. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 558. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).

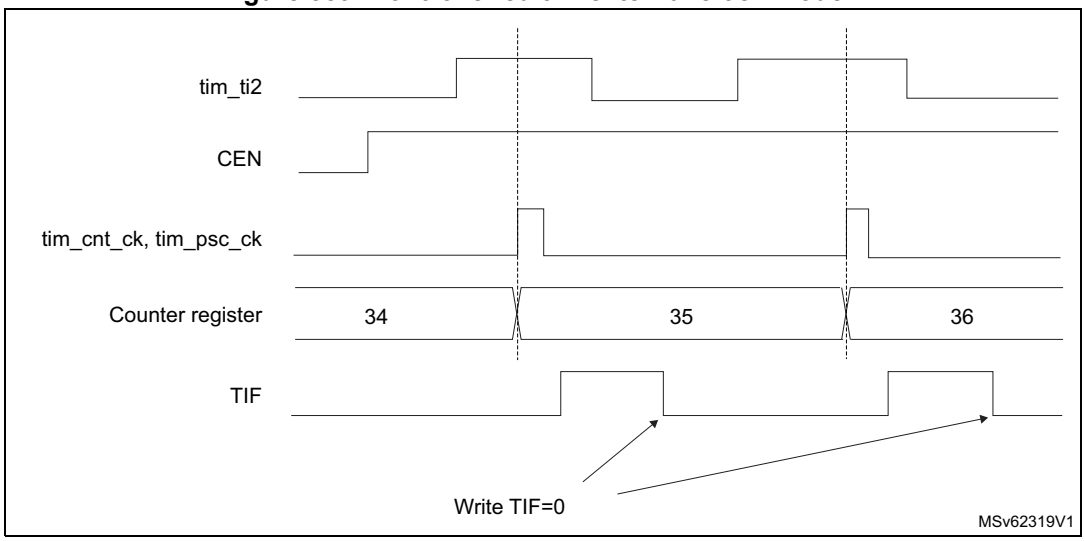
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 559. Control circuit in external clock mode 1



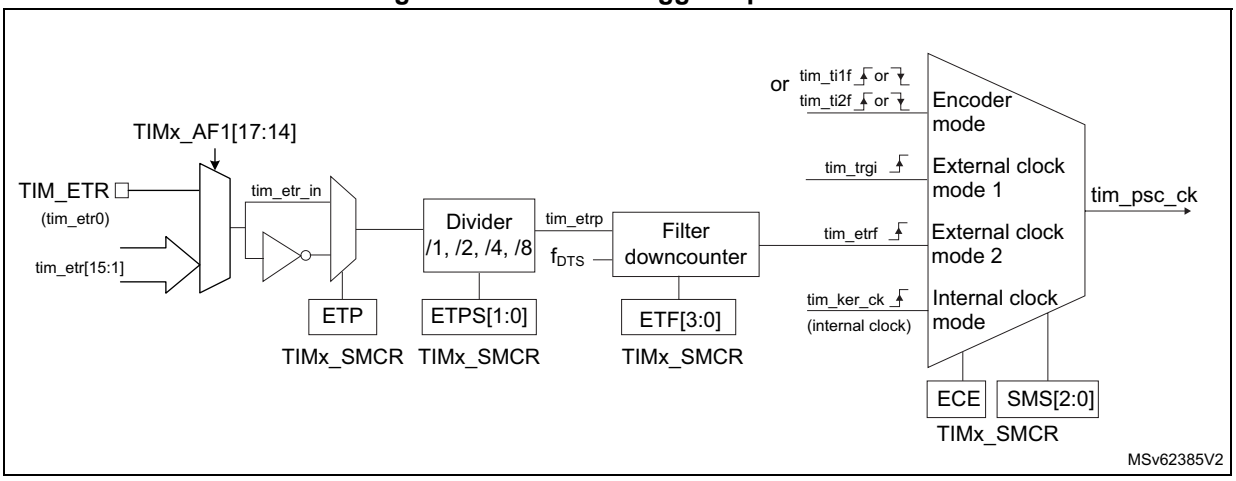
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 560 gives an overview of the external trigger input block.

Figure 560. External trigger input block



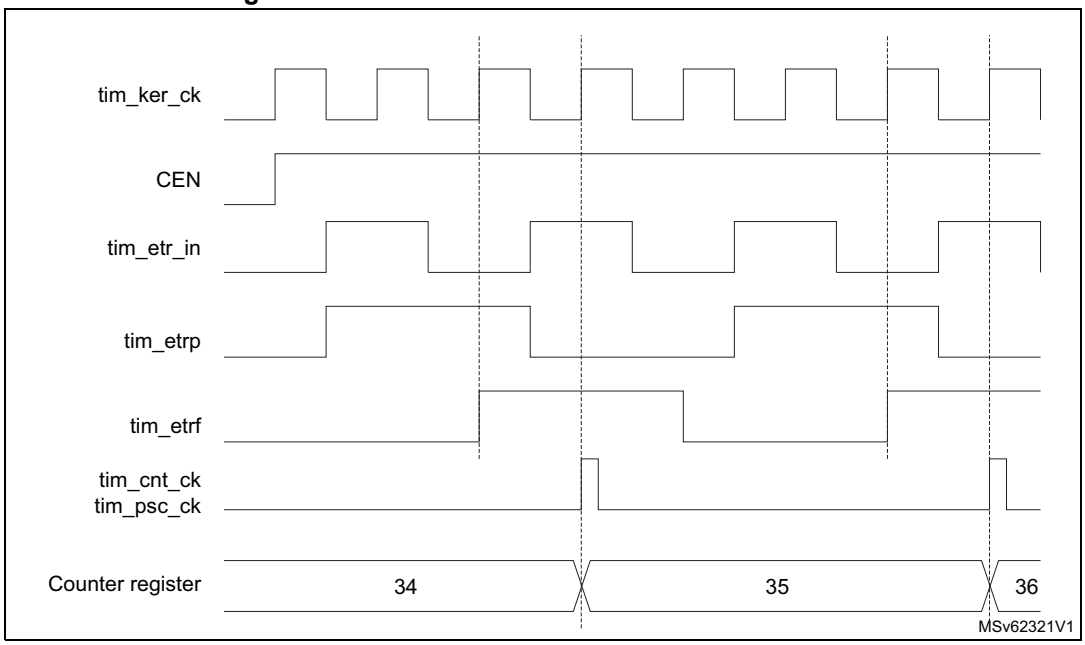
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 561. Control circuit in external clock mode 2



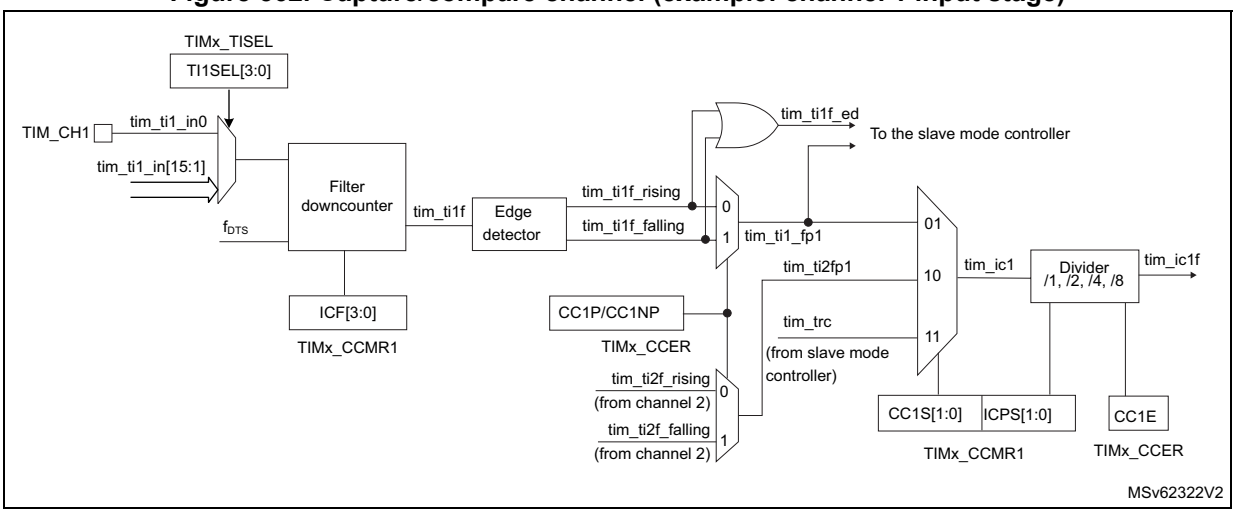
54.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 562. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 563. Capture/compare channel 1 main circuit

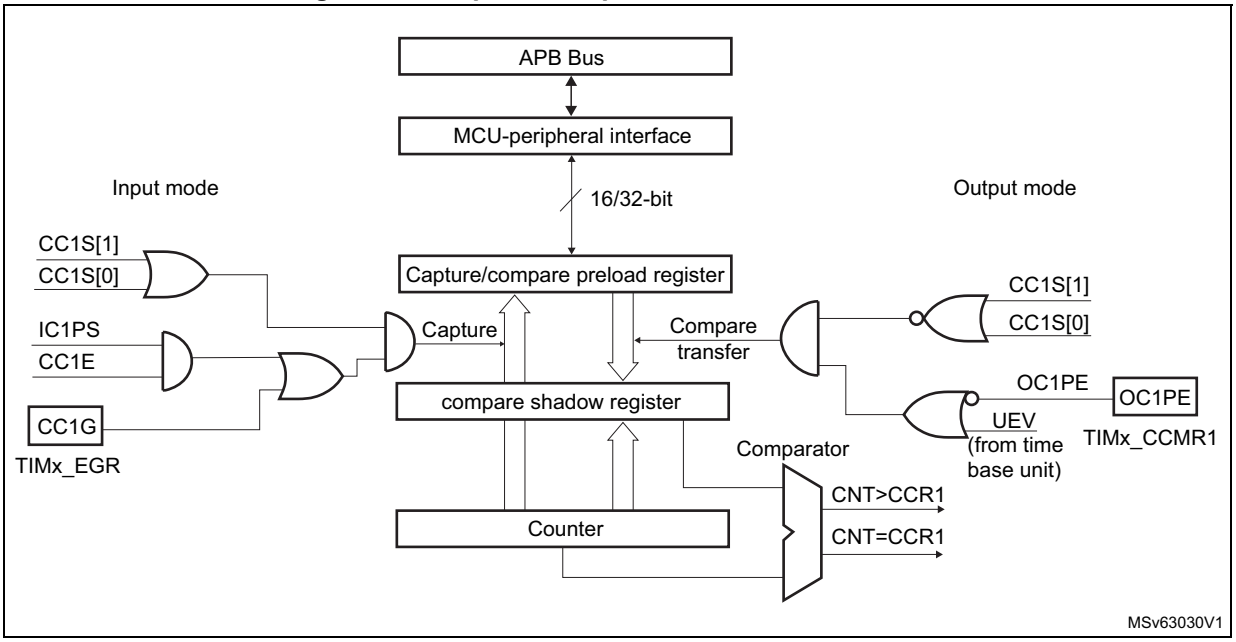
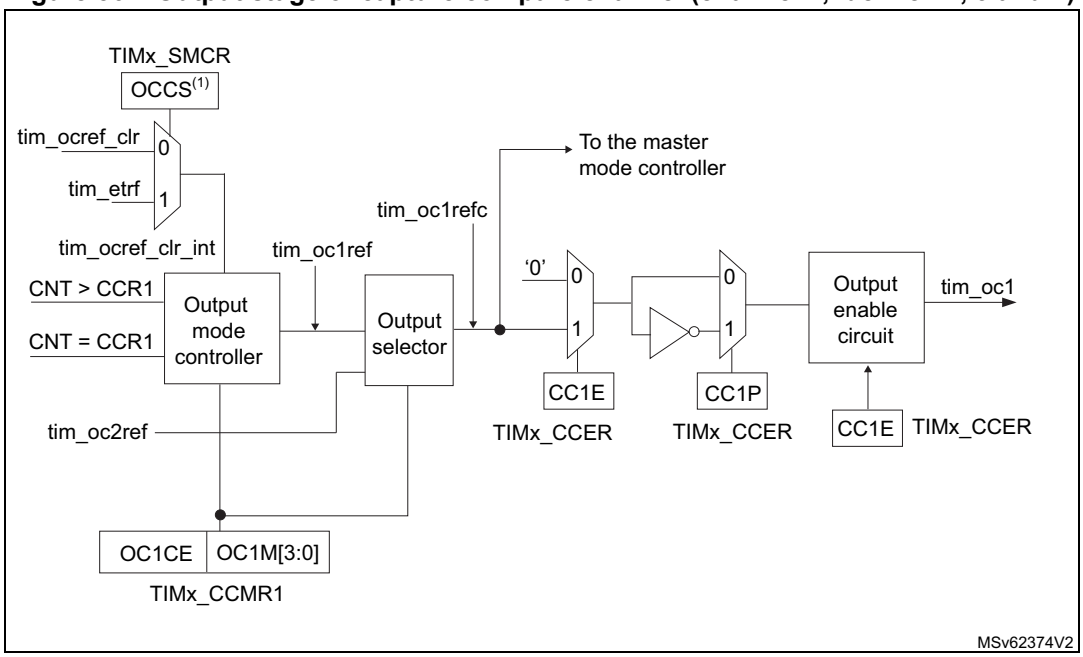


Figure 564. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

54.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

54.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

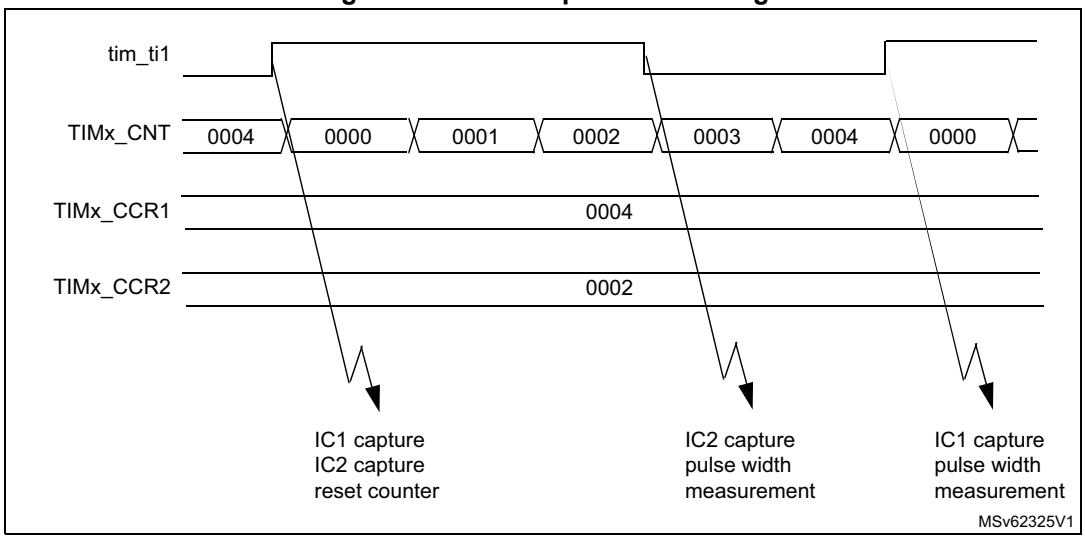
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 565. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

54.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

54.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

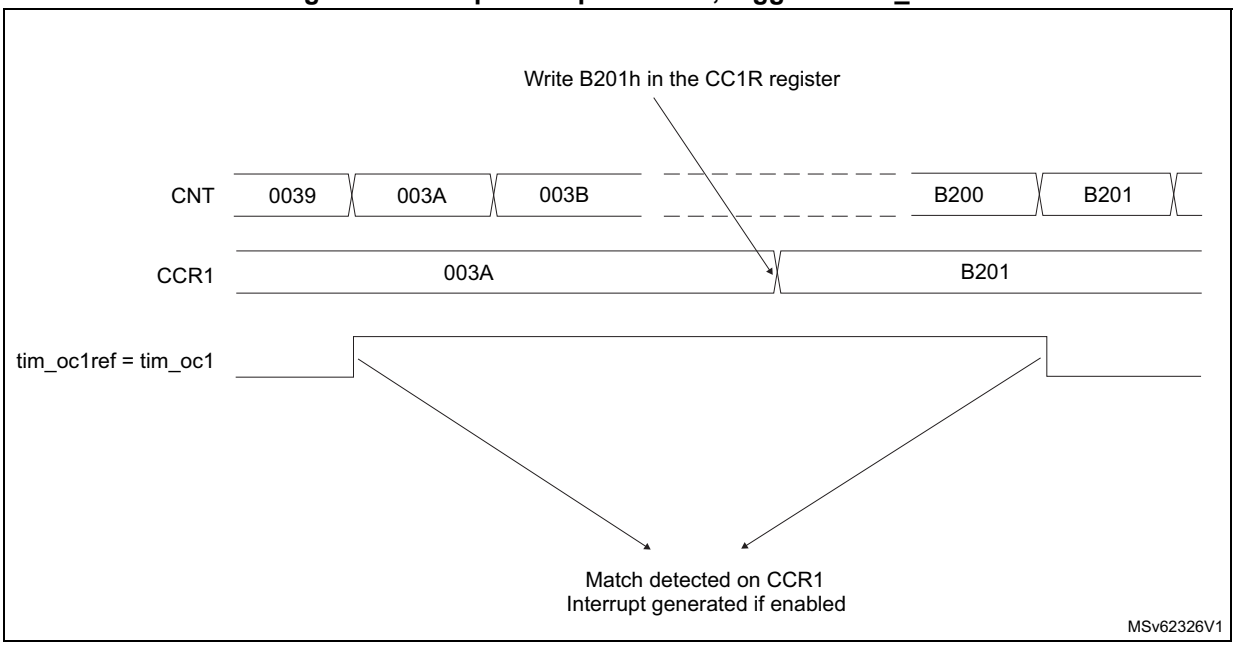
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 566*.

Figure 566. Output compare mode, toggle on tim_oc1



54.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

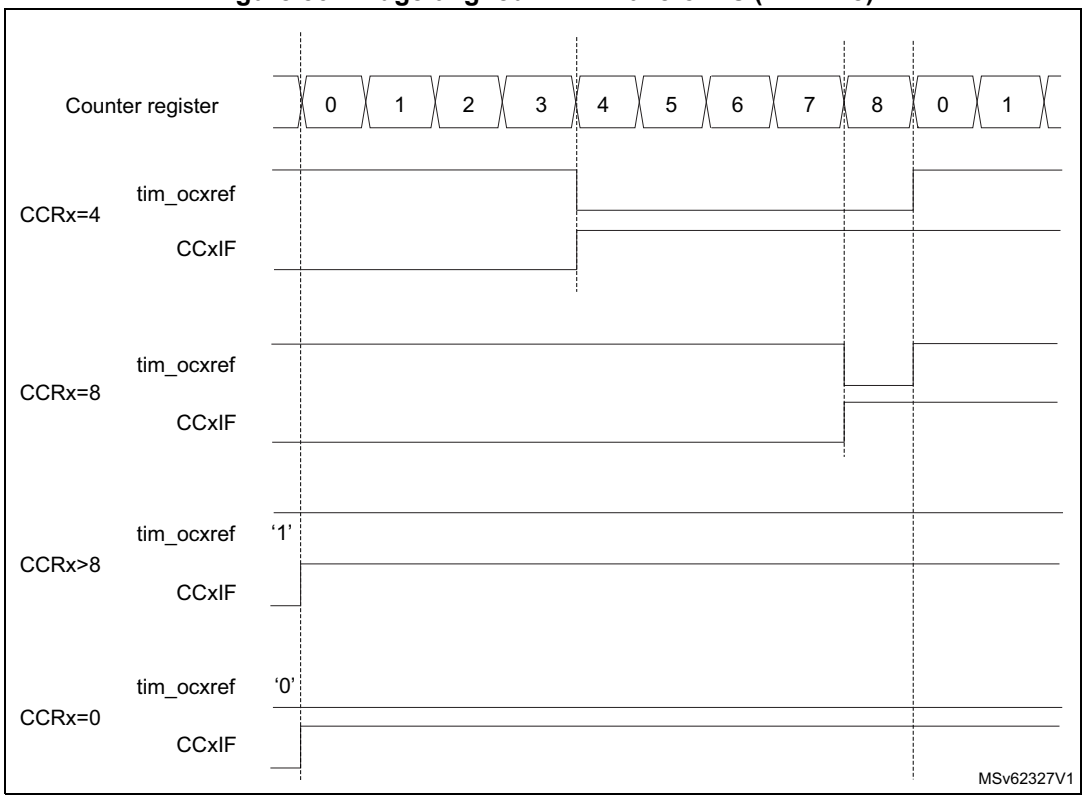
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 567* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 567. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

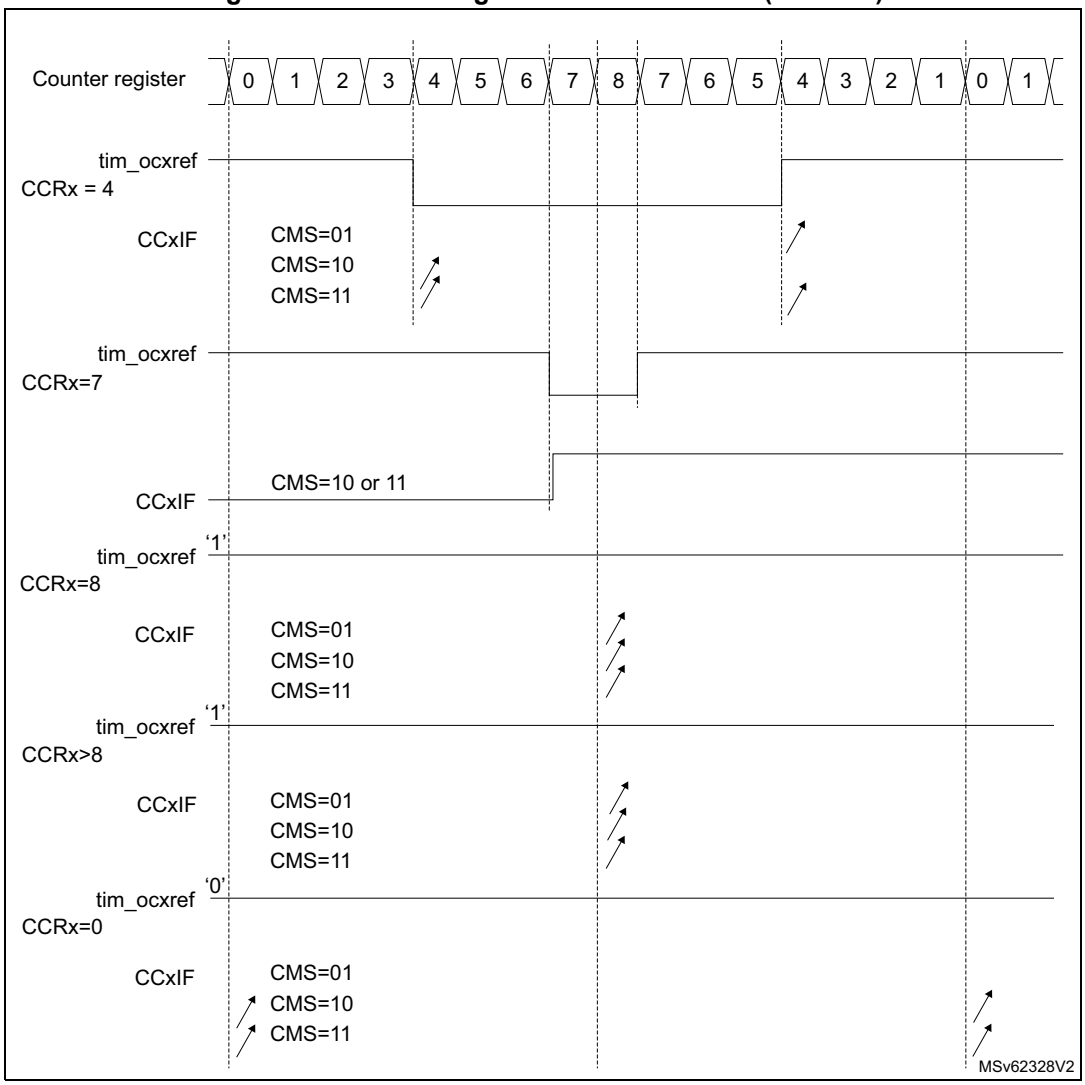
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 568 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 568. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

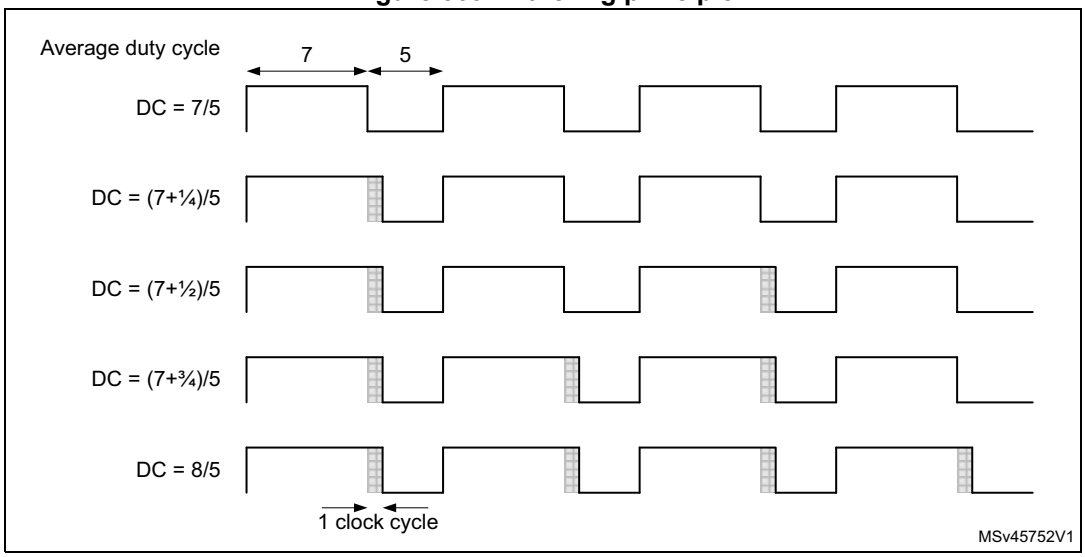
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 569* presents the dithering principle applied to four consecutive PWM cycles.

Figure 569. Dithering principle



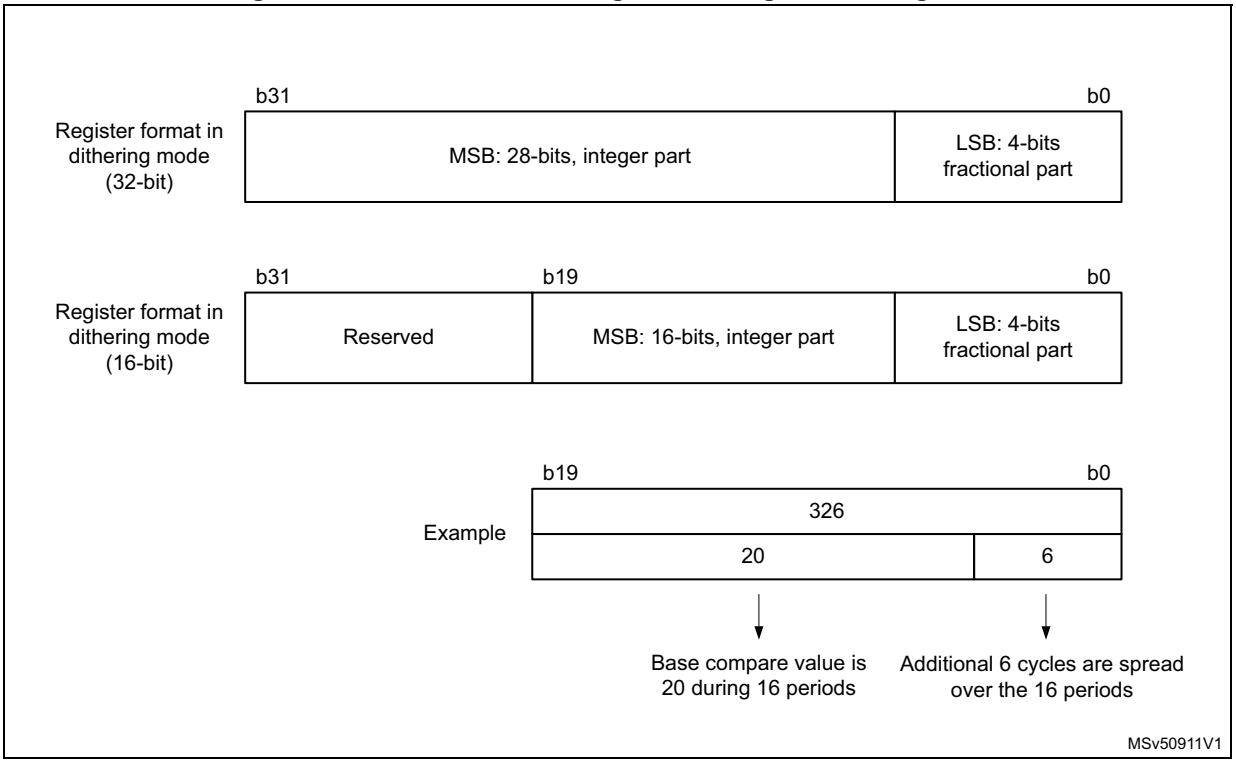
When the dithering mode is enabled, the register coding is changed as following (see *Figure 570* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 570. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 571* and *Figure 572*, the dithering mode is used to increase the PWM resolution.

Figure 571. PWM resolution vs frequency (16-bit mode)

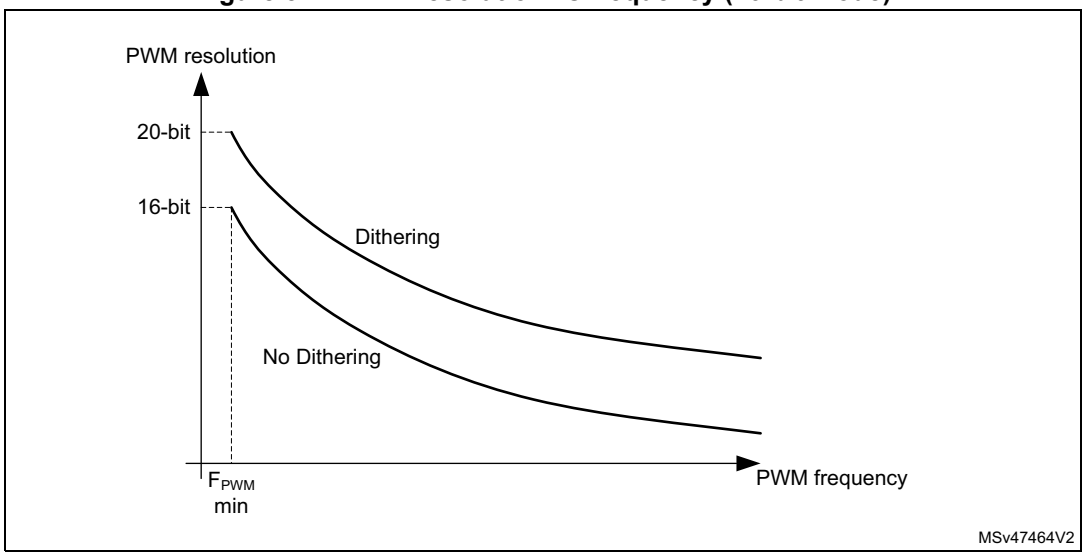
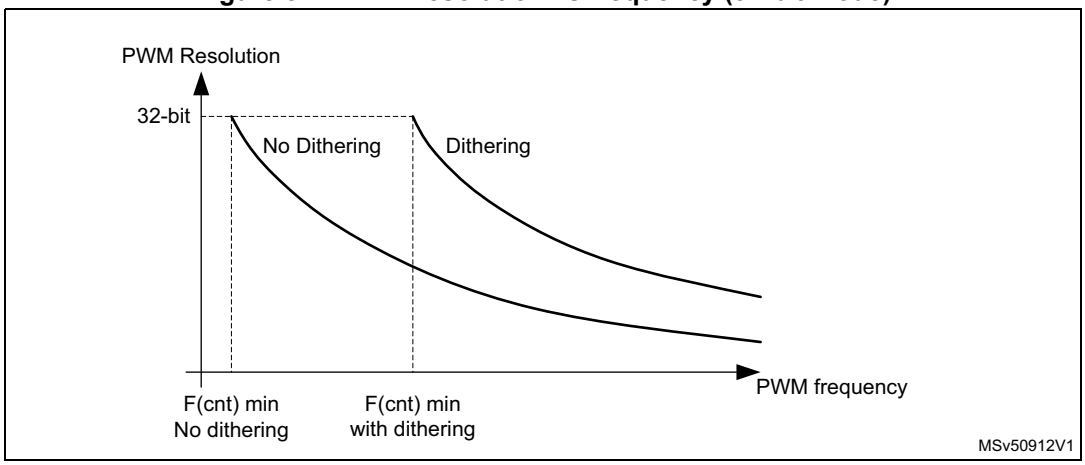
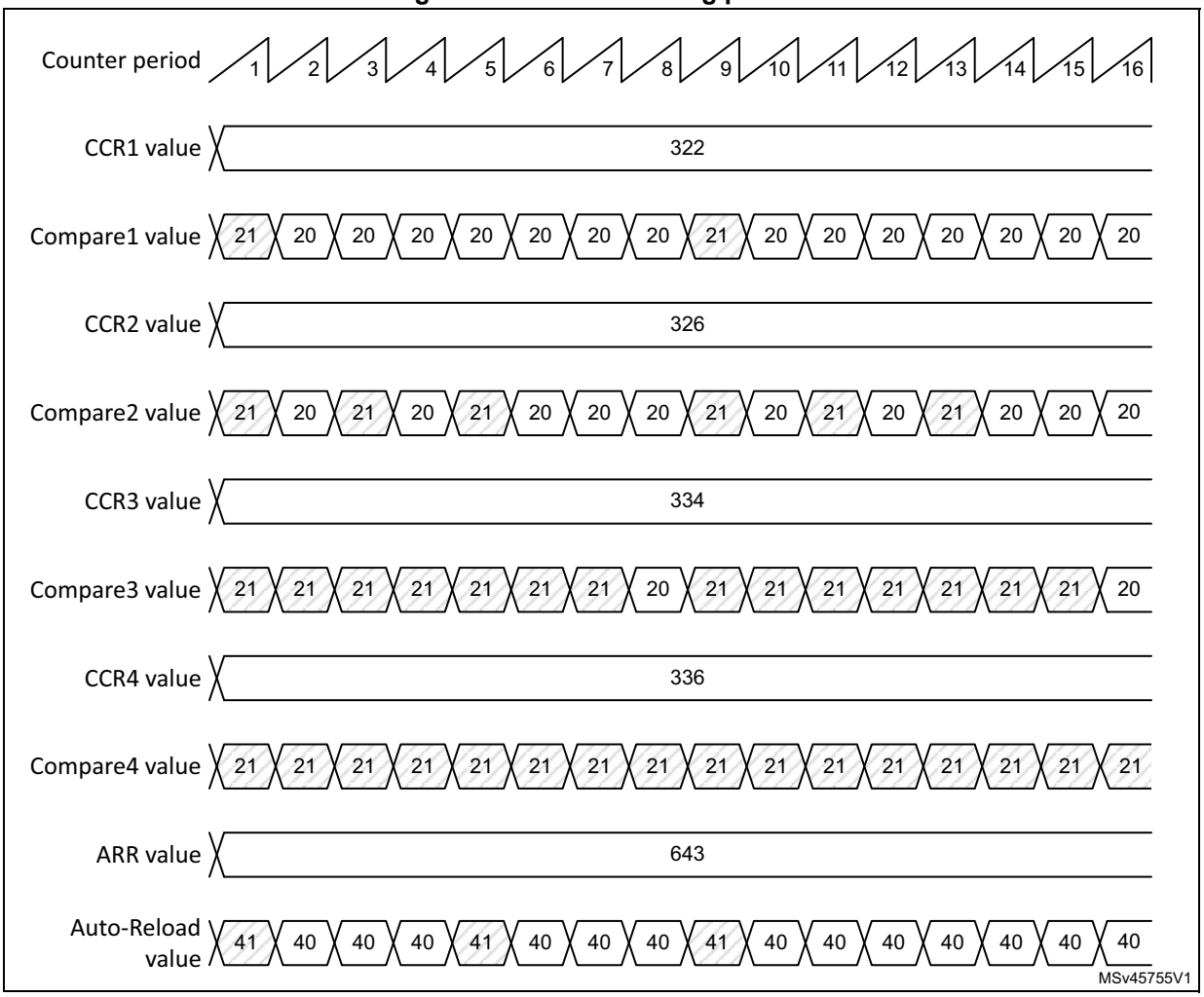


Figure 572. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 573*.

Figure 573. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 549*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 549. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 574*.

Figure 574. Dithering effect on duty cycle in center-aligned PWM mode

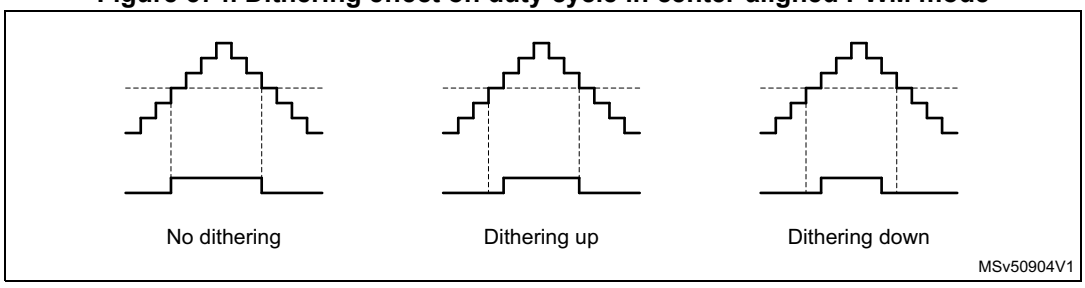


Table 550 shows how the dithering pattern is added in center-aligned PWM mode.

Table 550. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

54.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

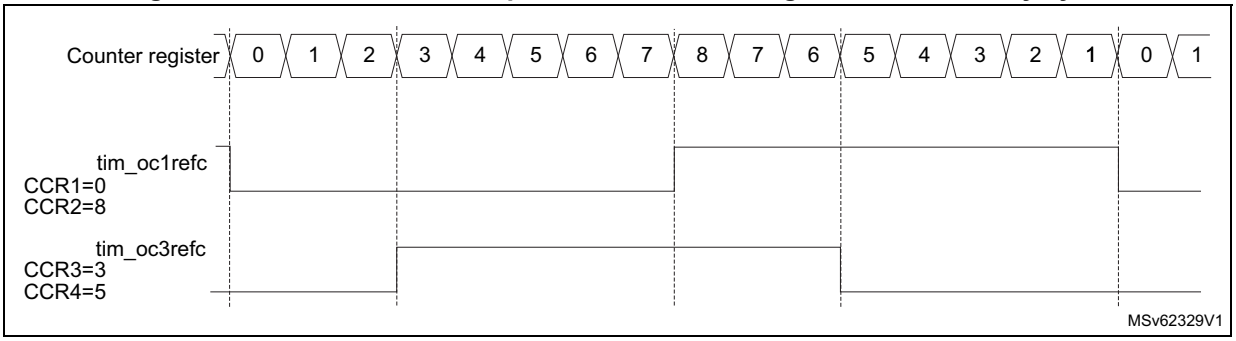
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 575 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 575. Generation of two phase-shifted PWM signals with 50% duty cycle



54.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

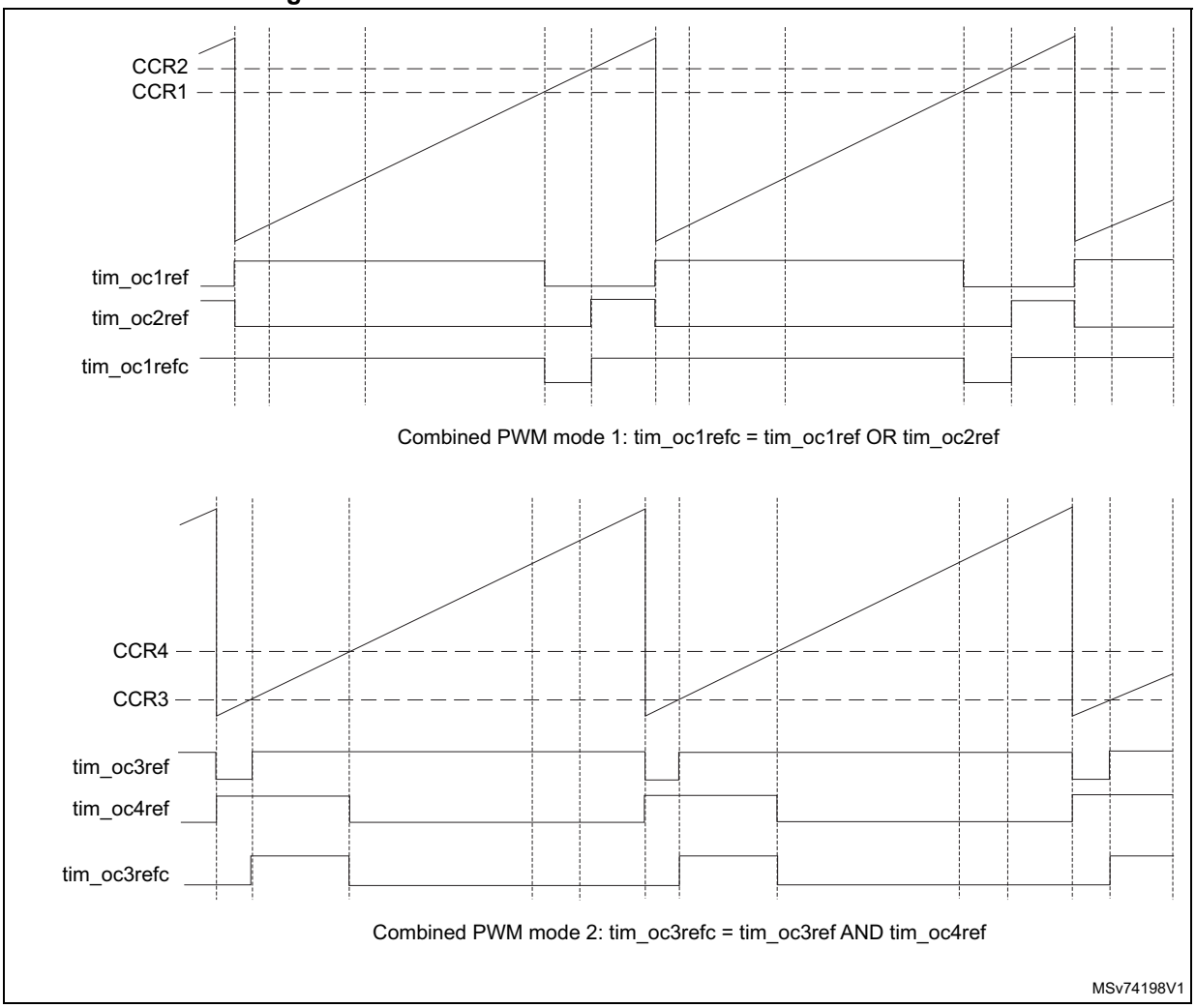
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 576 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 576. Combined PWM mode on channels 1 and 3



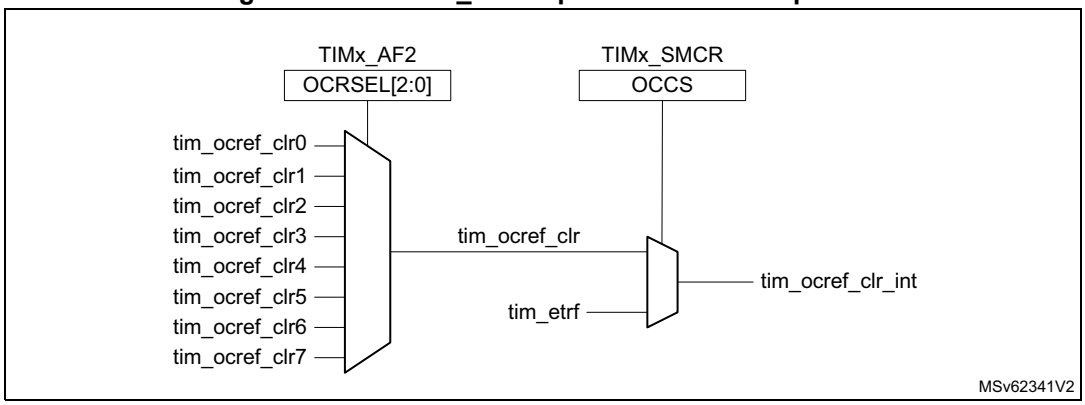
54.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 577*.

Figure 577. OCREF_CLR input selection multiplexer



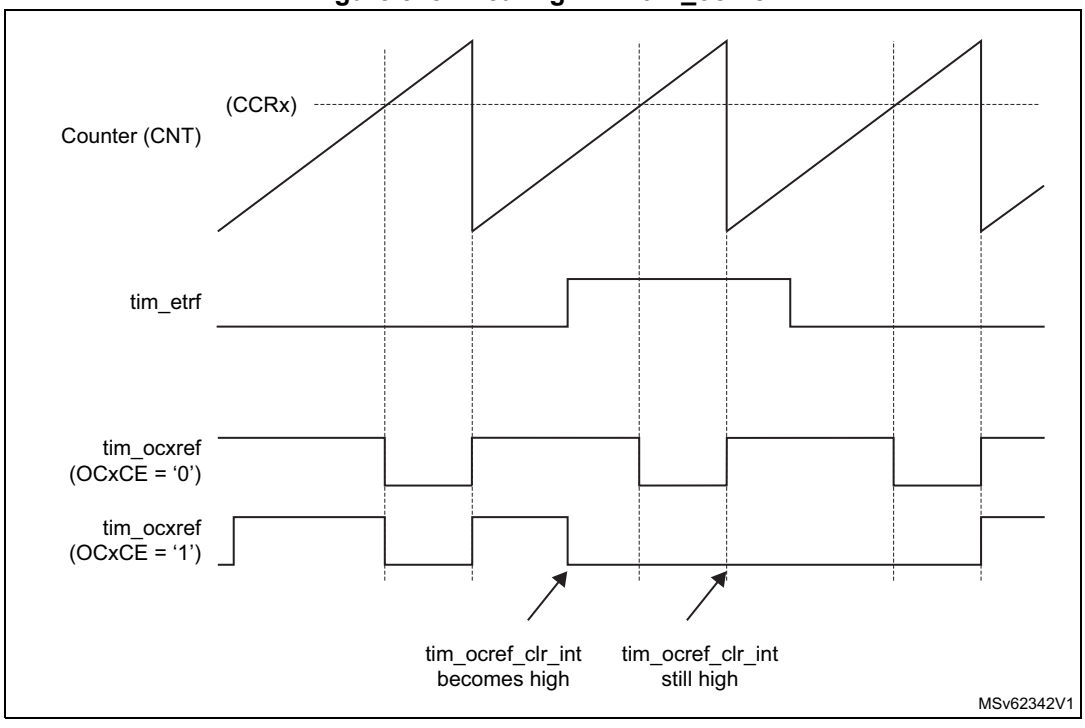
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIMx_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 578 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 578. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

54.4.15 One-pulse mode

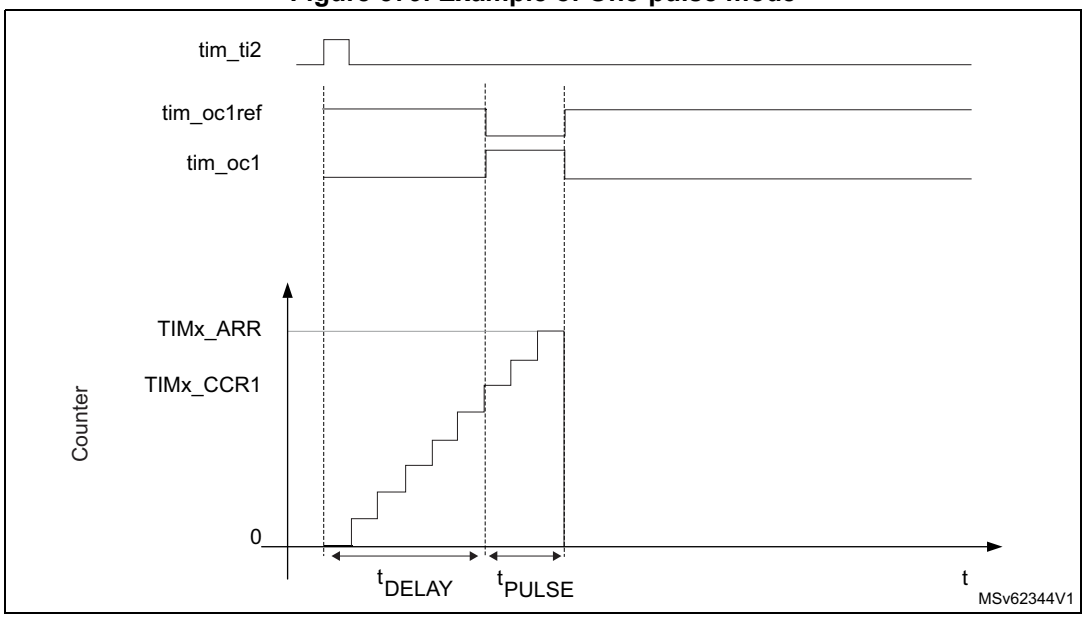
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 579. Example of One-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

54.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 54.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

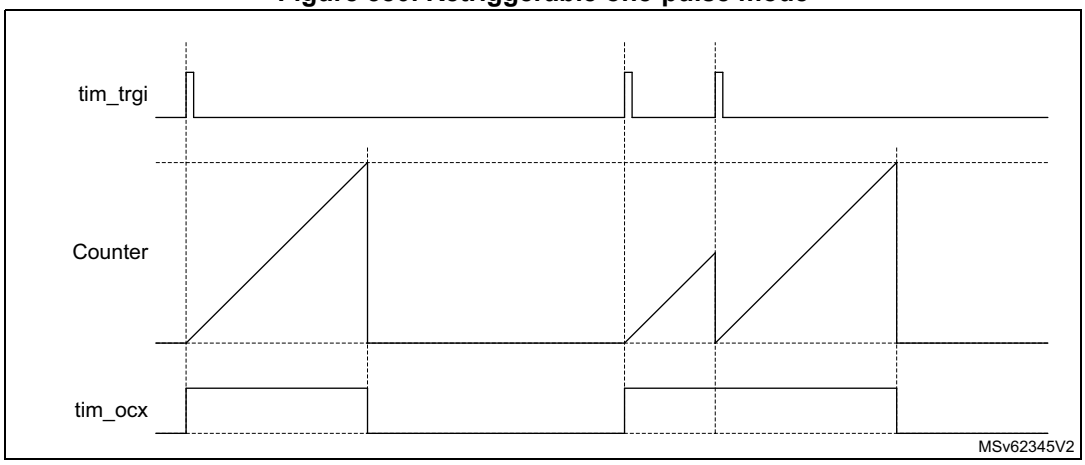
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 580. Retriggerable one-pulse mode



54.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 581*.

Figure 581. Pulse generator circuitry

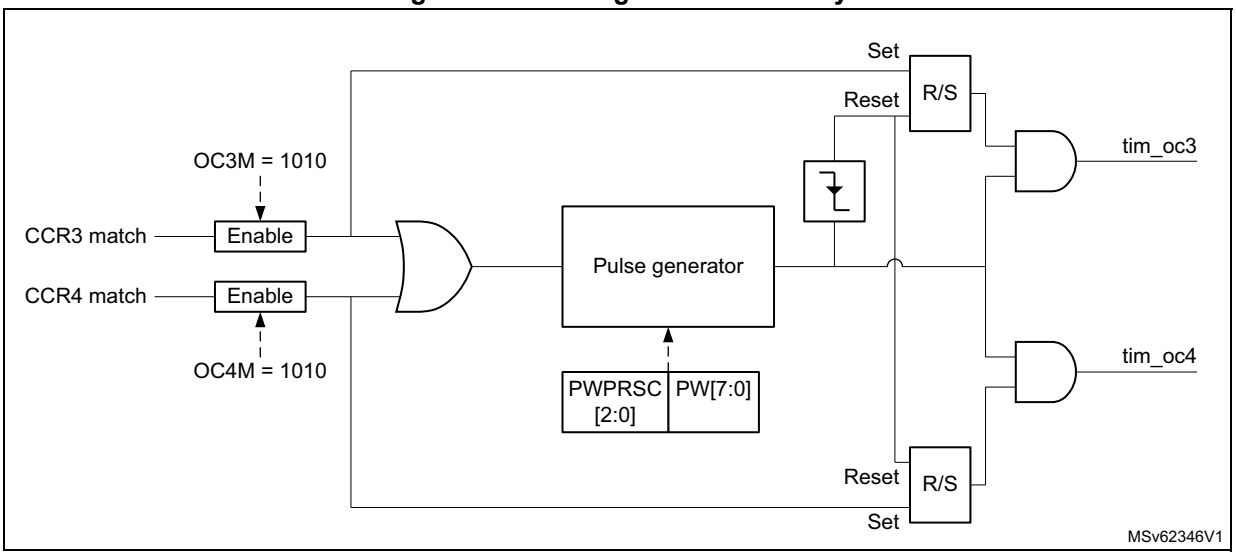
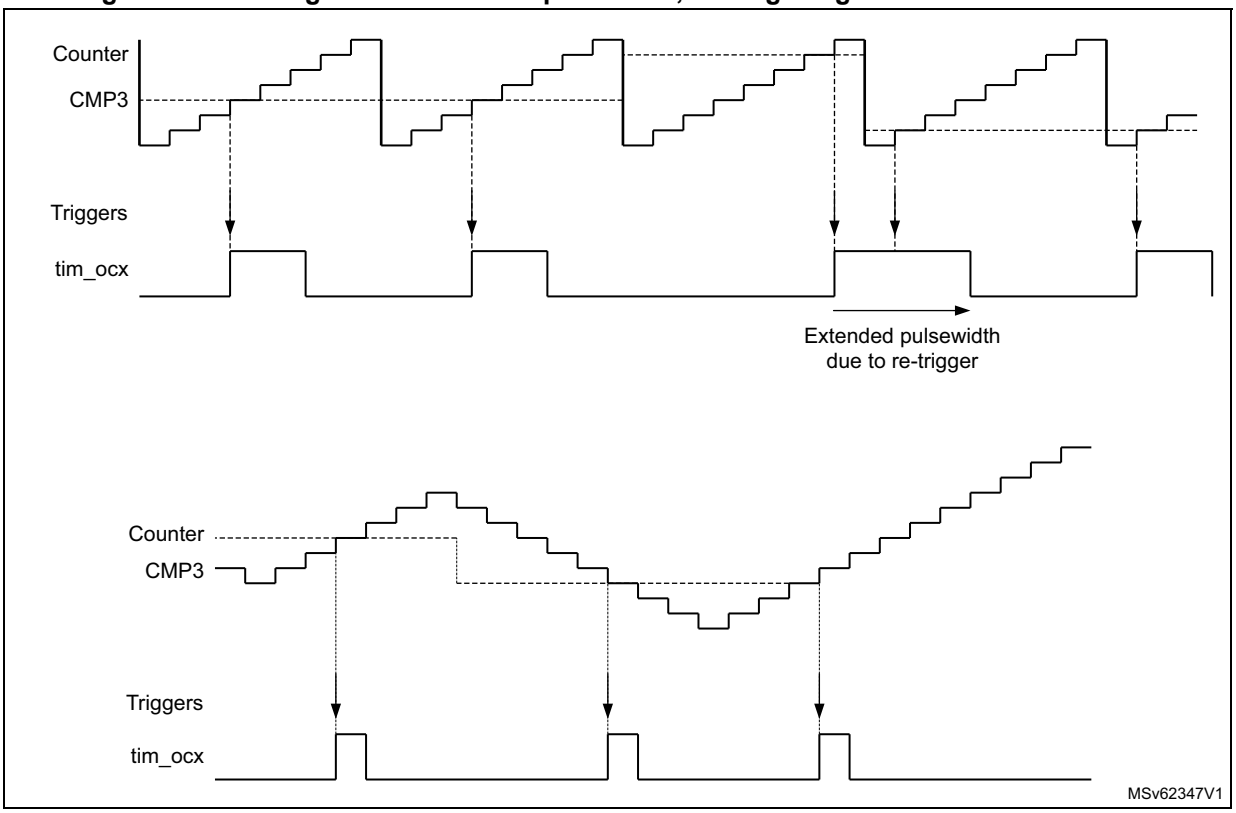


Figure 582 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 582. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

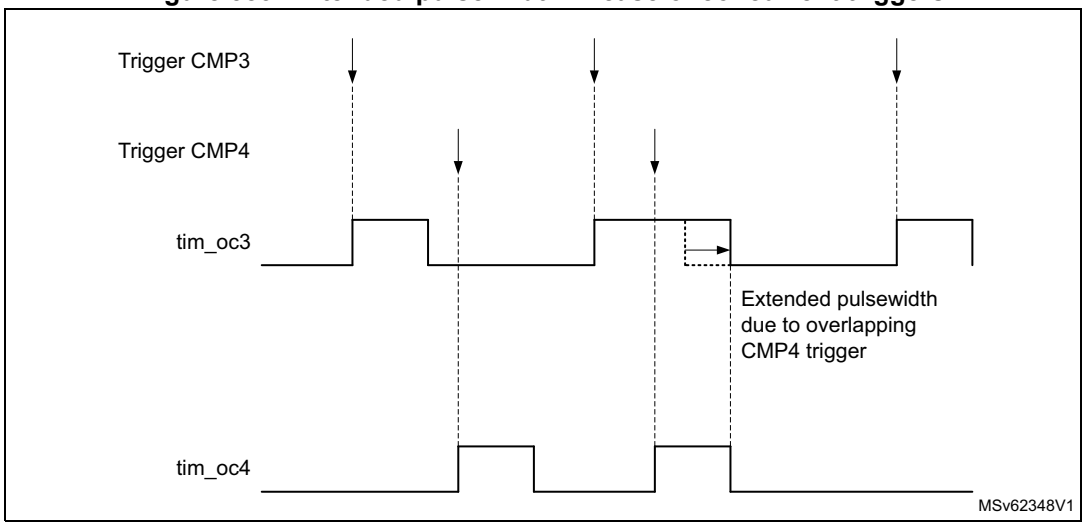
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 583).

Figure 583. Extended pulse width in case of concurrent triggers



54.4.18 Encoder interface mode

Quadrature encoder

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 551*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or

tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 551. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 584 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 584. Example of counter operation in encoder interface mode

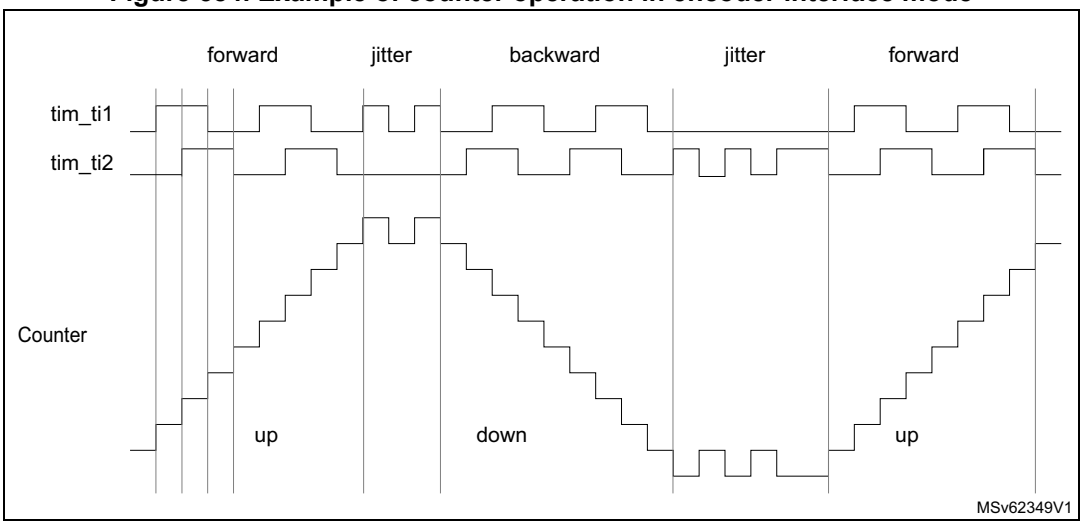


Figure 585 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 585. Example of encoder interface mode with tim_ti1fp1 polarity inverted

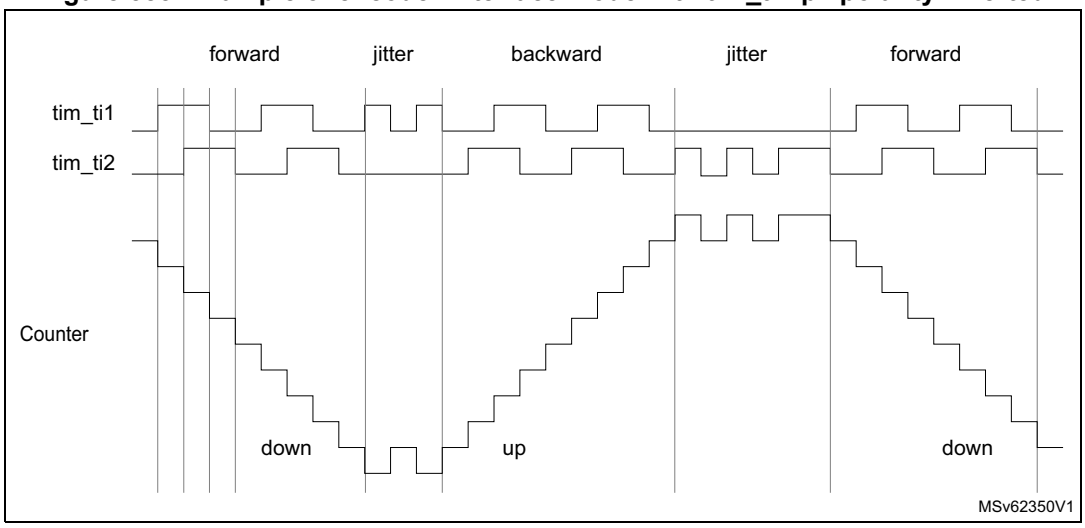
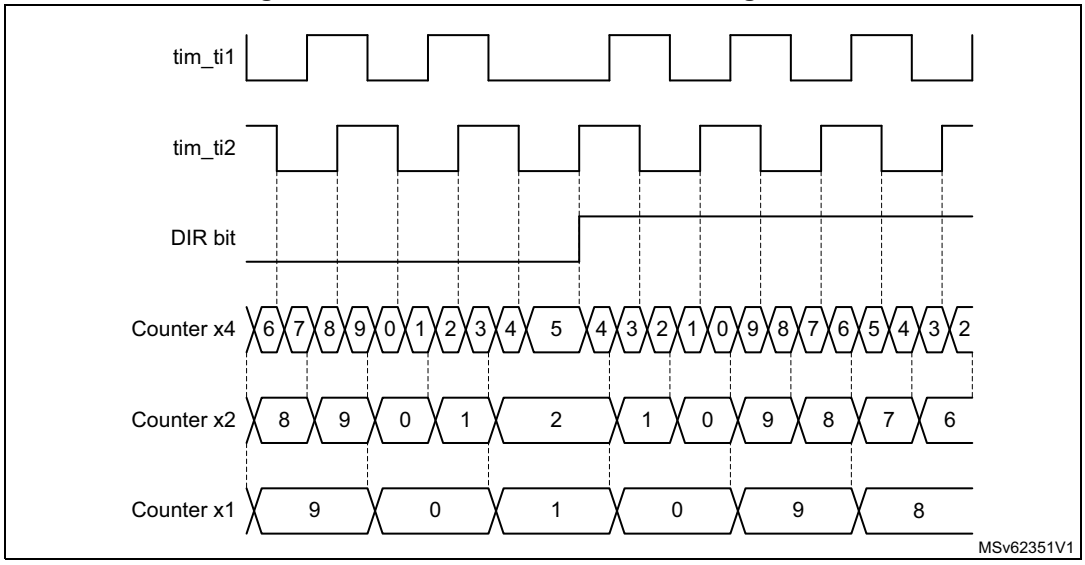


Figure 586 shows the timer counter value during a speed reversal, for various counting modes.

Figure 586. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

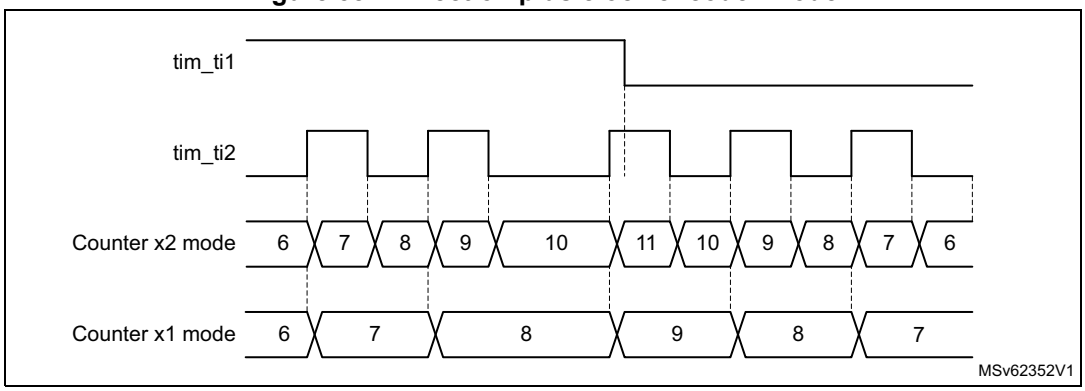
In the "clock plus direction" mode shown on *Figure 587*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 587. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 588*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 588*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 589*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 588*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 589*).

Figure 588. Directional clock encoder mode (CC1P = CC2P = 0)

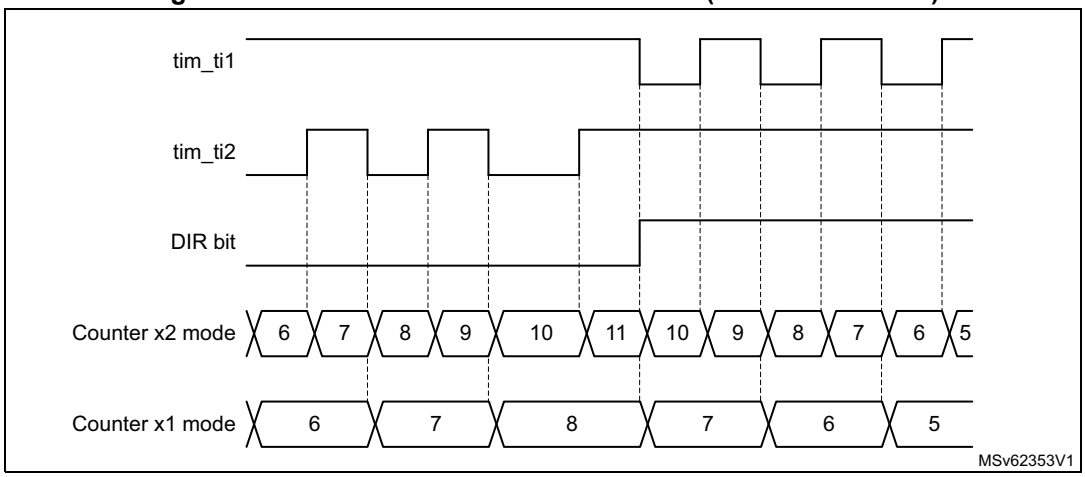


Figure 589. Directional clock encoder mode (CC1P = CC2P = 1)

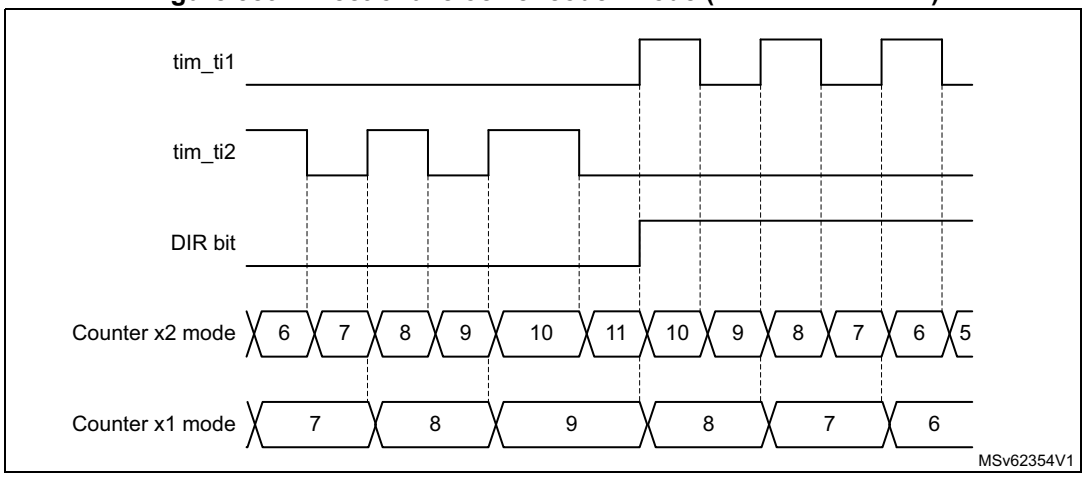


Table 552 details how the directional clock mode operates, for any input transition.

Table 552. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

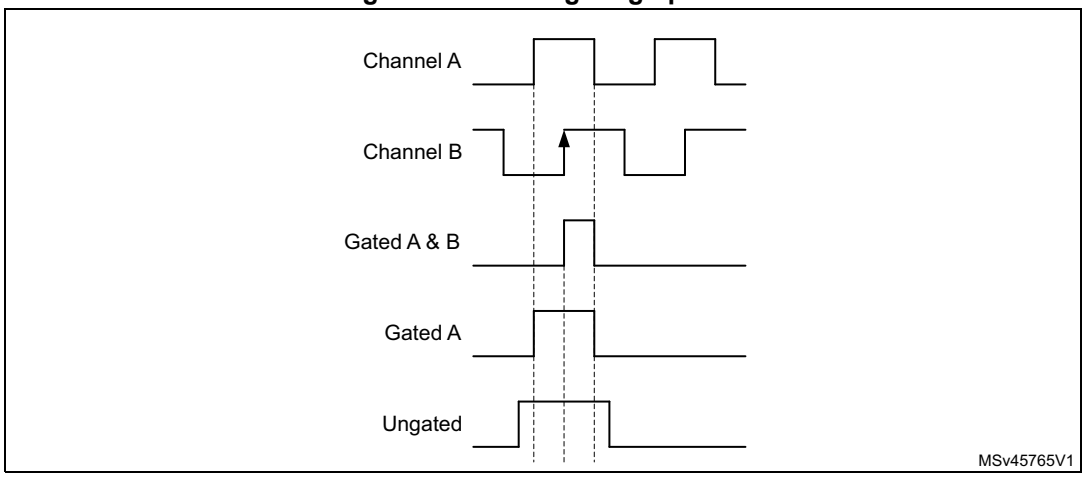
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 590*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

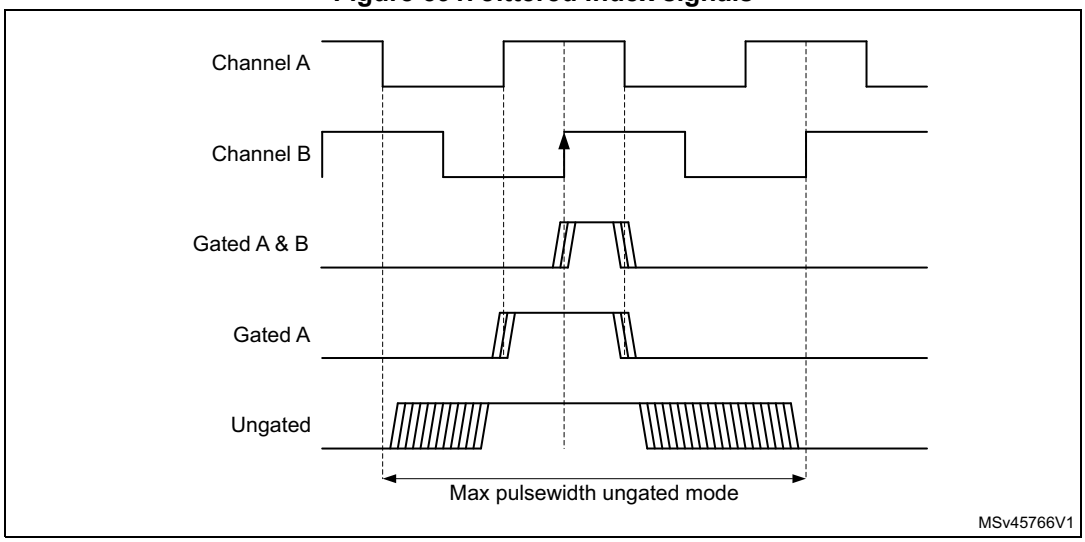
Figure 590. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 591*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 591. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 592* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 592. Index generation for IPOS[1:0] = 11

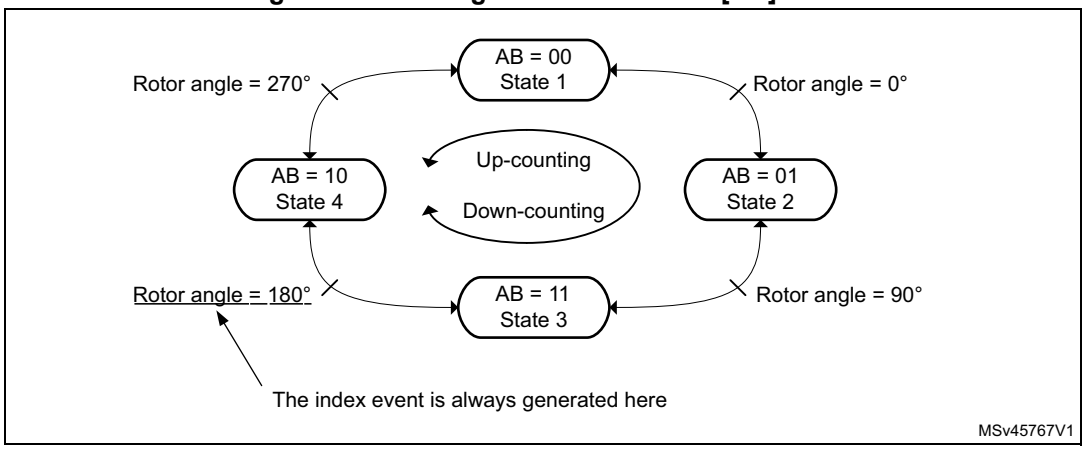


Figure 593 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 593. Counter reading with index gated on channel A (IPOS[1:0] = 11)

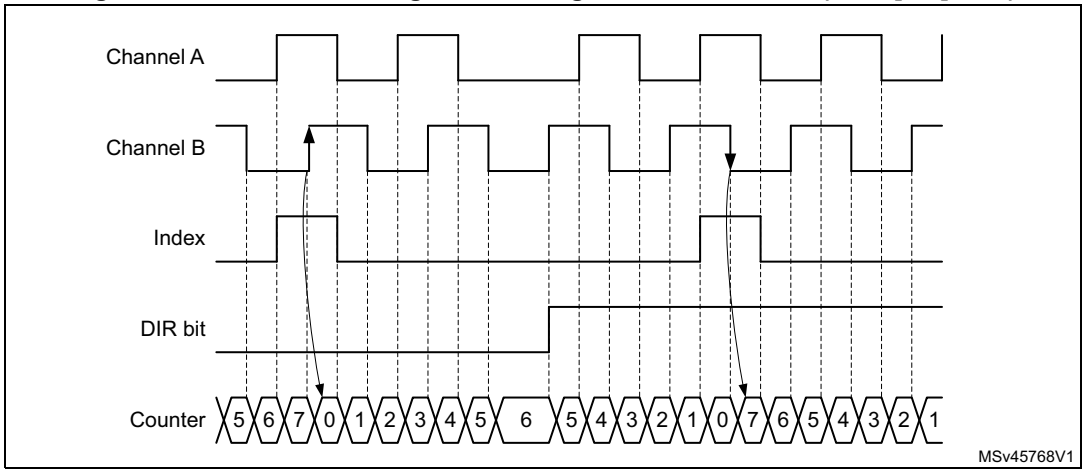


Figure 594 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 594. Counter reading with index ungated (IPOS[1:0] = 00)

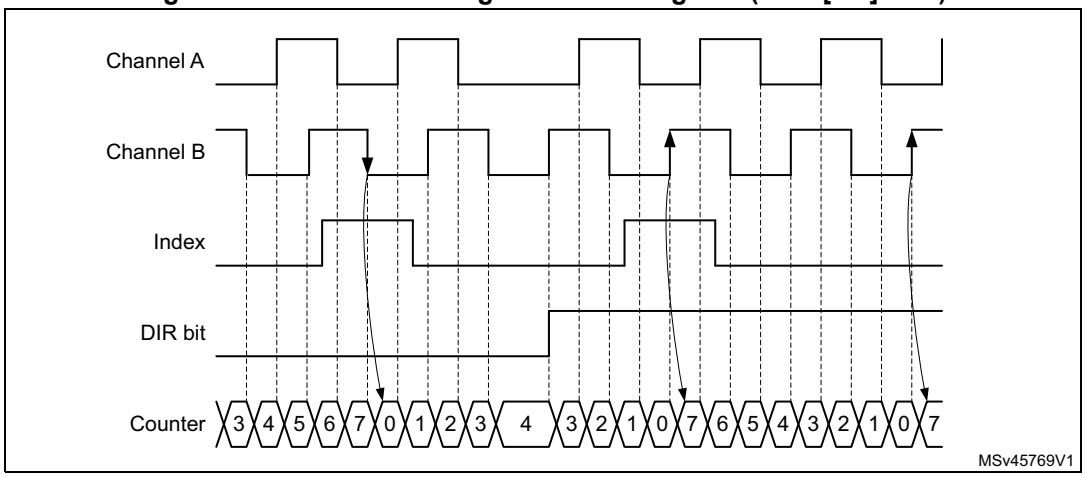


Figure 595 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 595. Counter reading with index gated on channel A and B

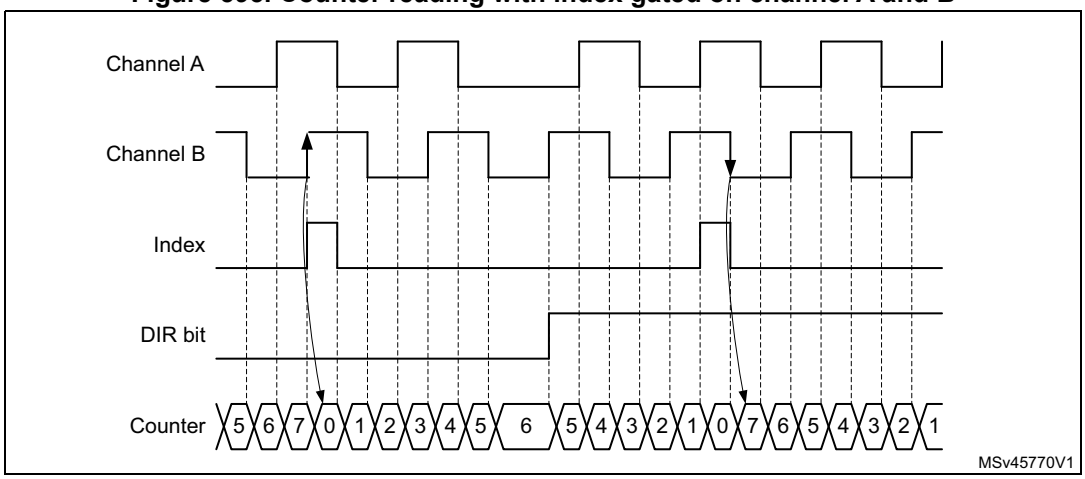


Figure 596 and *Figure 597* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 596. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

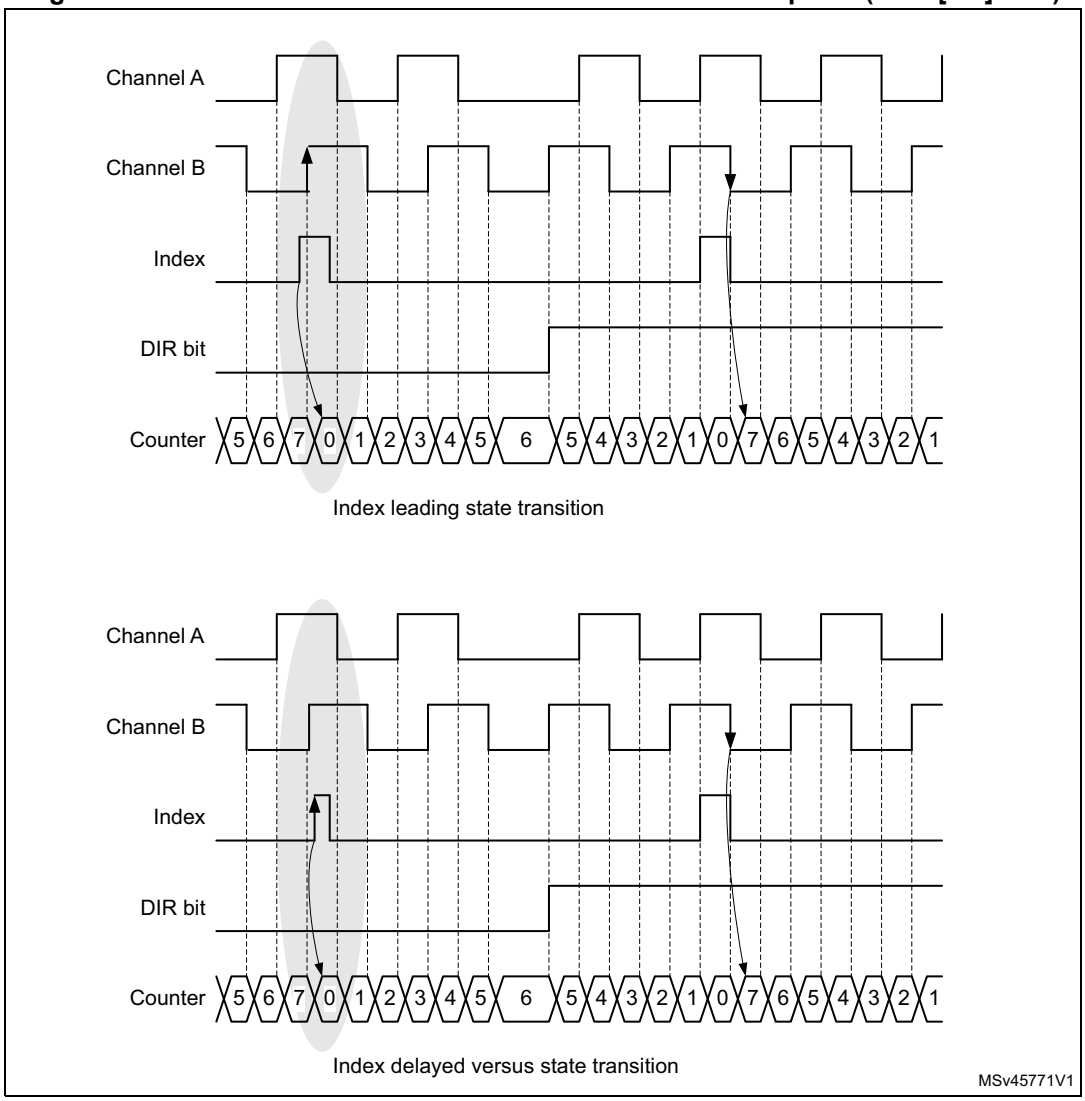


Figure 597. Counter reset Narrow index pulse (closer view, ARR = 0x07)

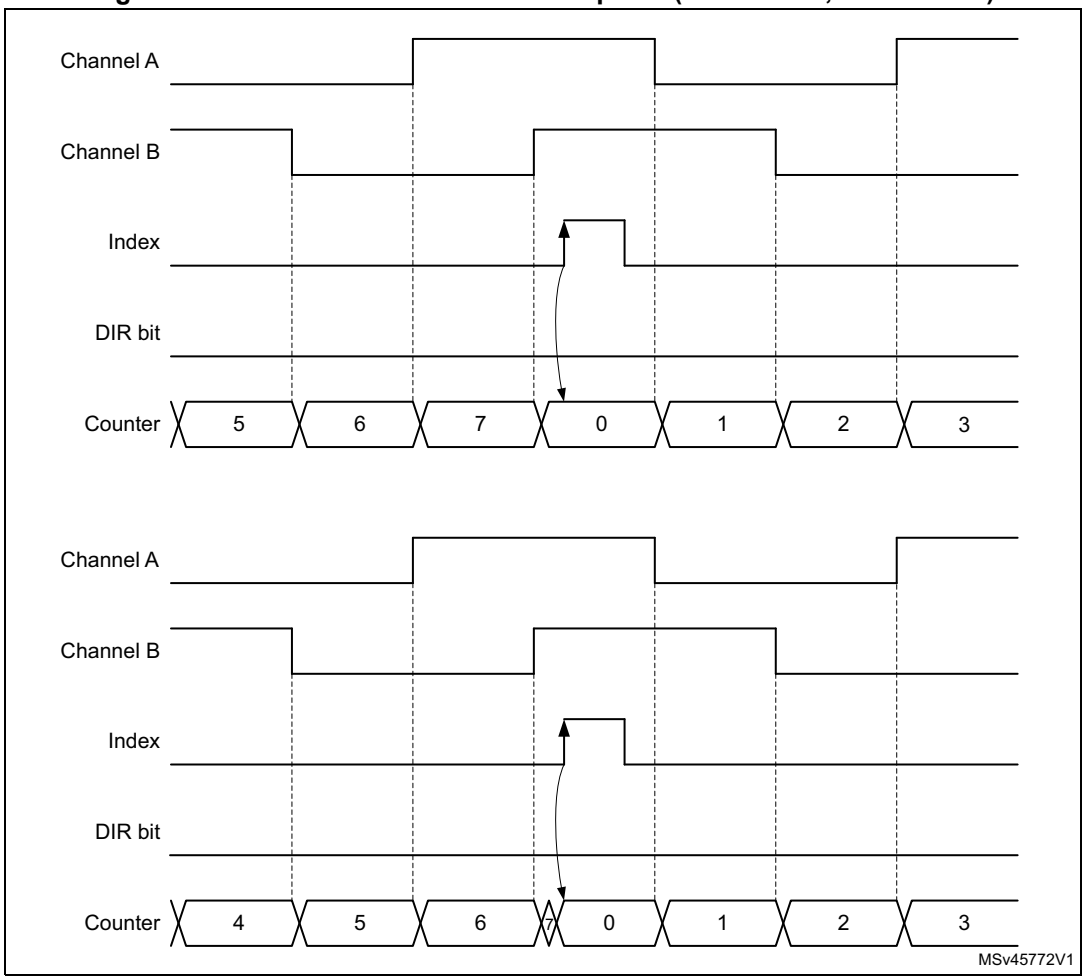
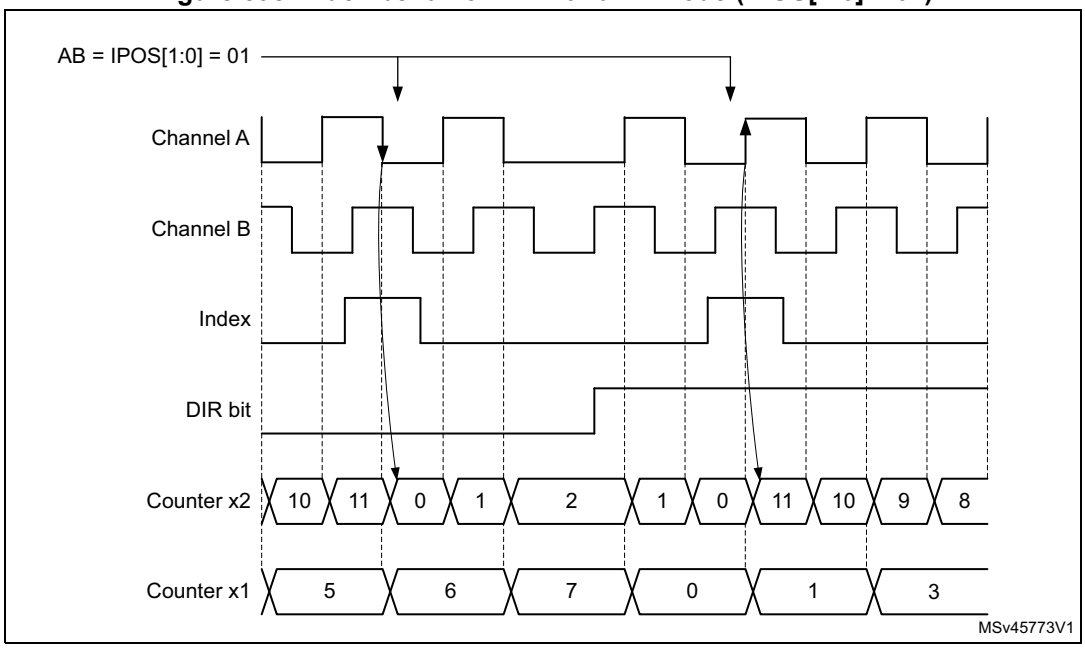


Figure 598 shows how the index is managed in x1 and x2 modes.

Figure 598. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

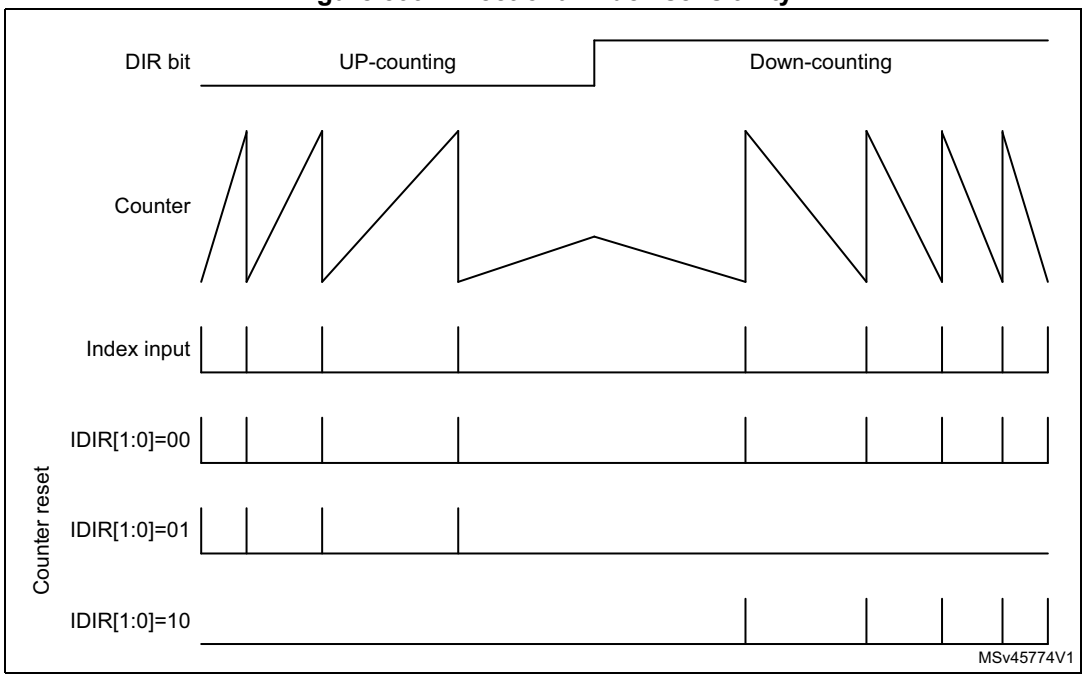


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 599 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

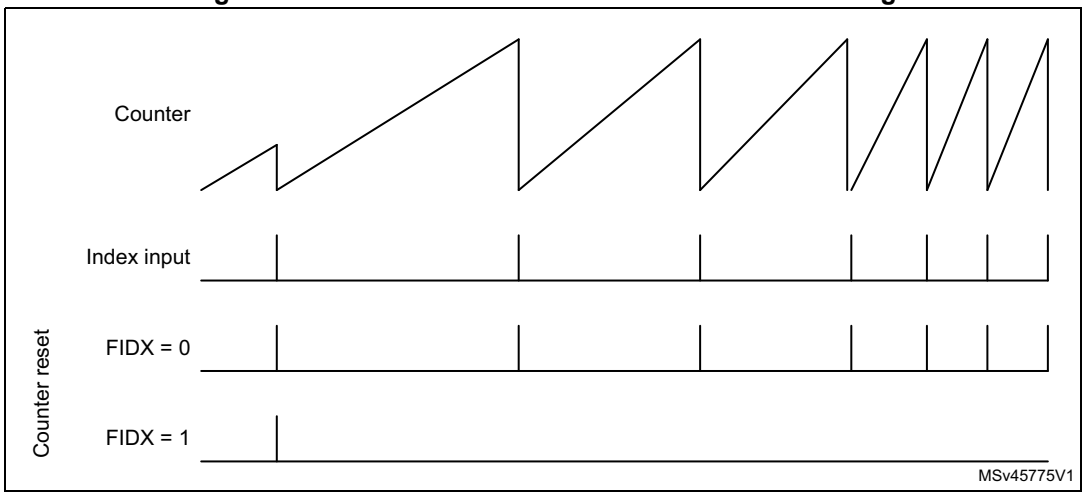
Figure 599. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 600*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 600. Counter reset as function of FIDX bit setting



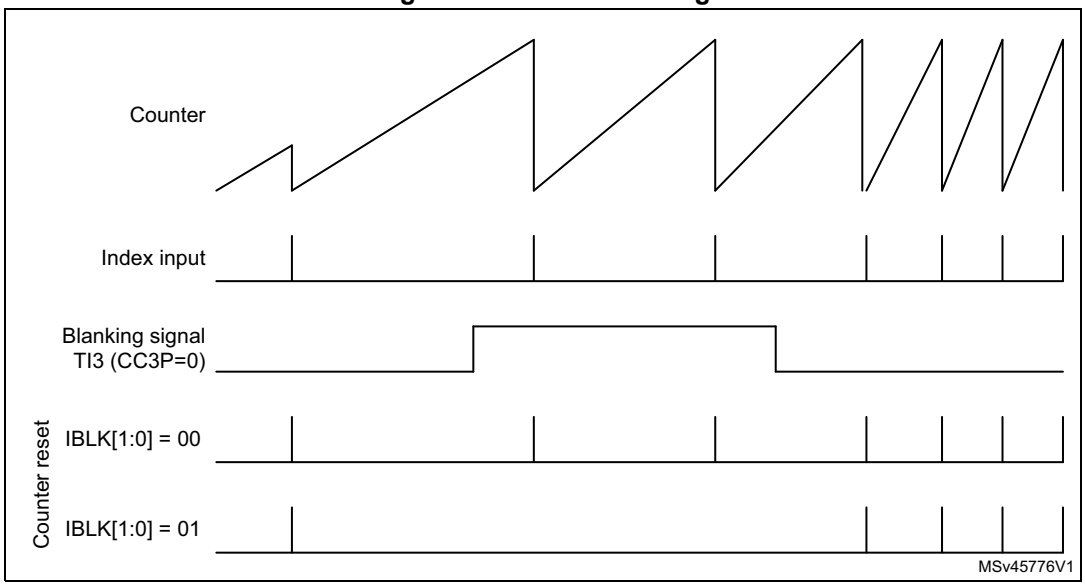
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 601*.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 601. Index blanking



Index management in nonquadrature mode

Figure 602 and *Figure 603* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 602. Index behavior in clock + direction mode, IPOS[0] = 1

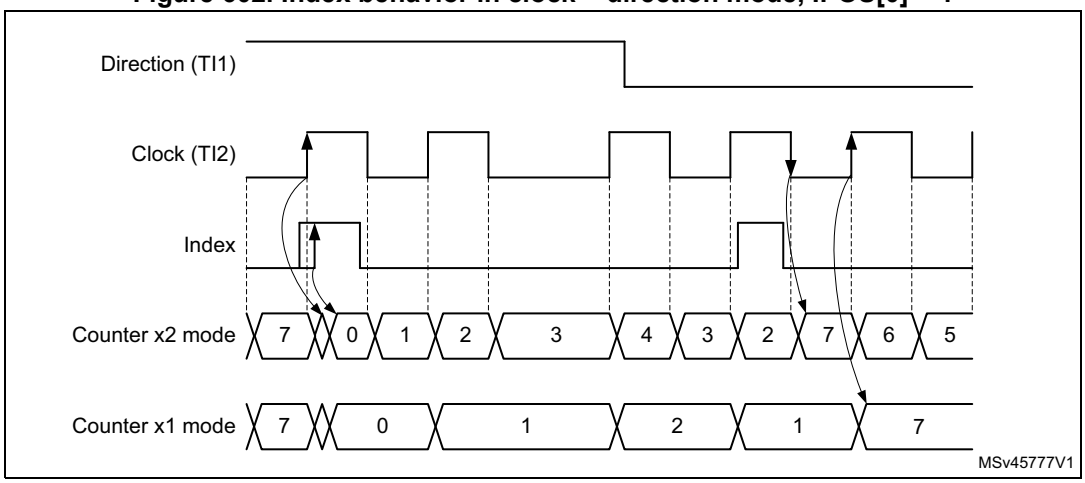
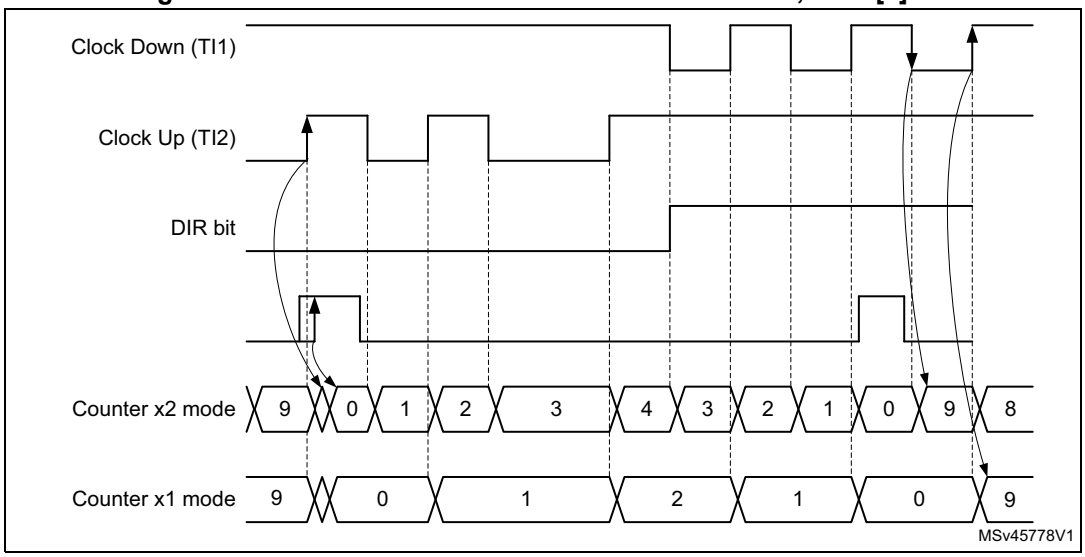


Figure 603. Index behavior in directional clock mode, IPOS[0] = 1

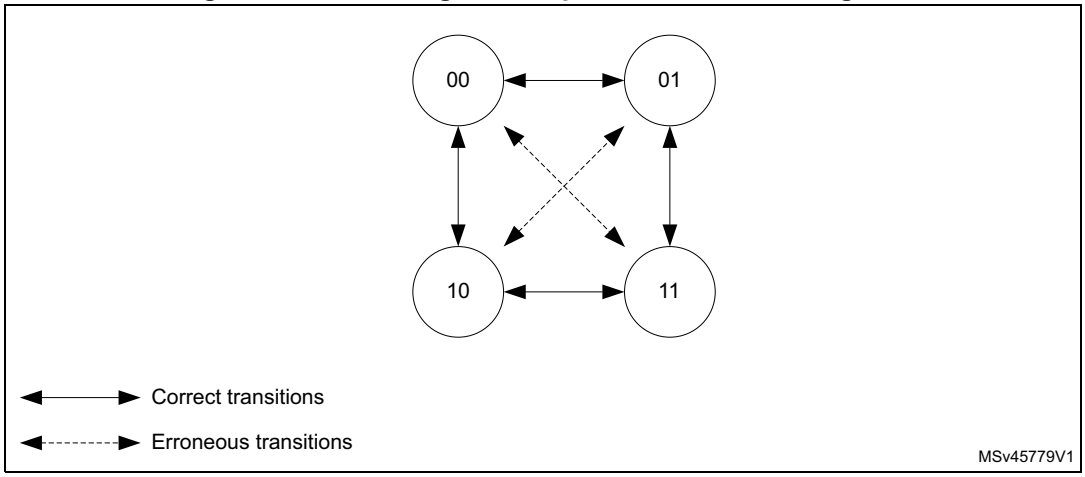


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 604*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 604. State diagram for quadrature encoded signals



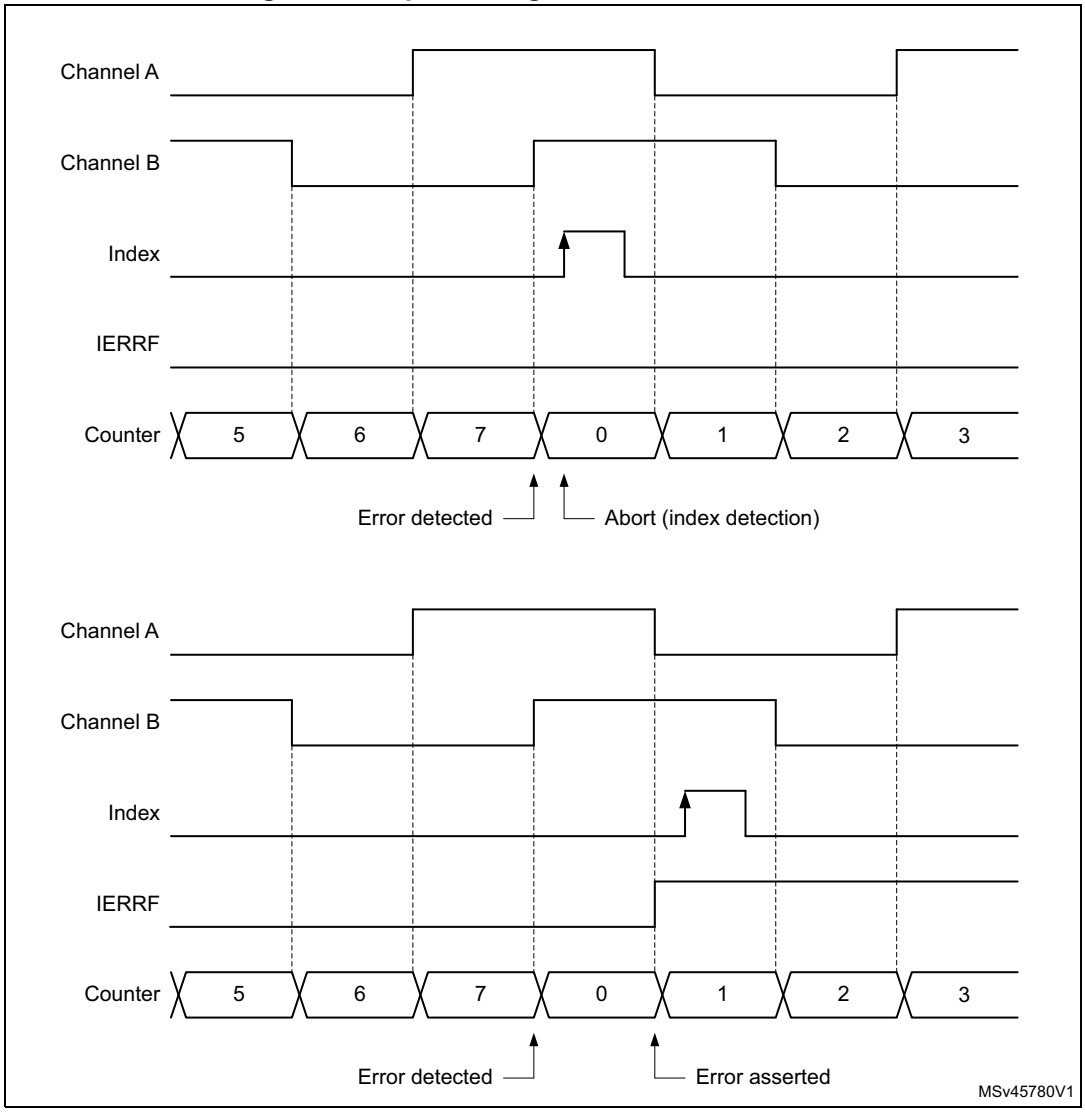
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMx_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

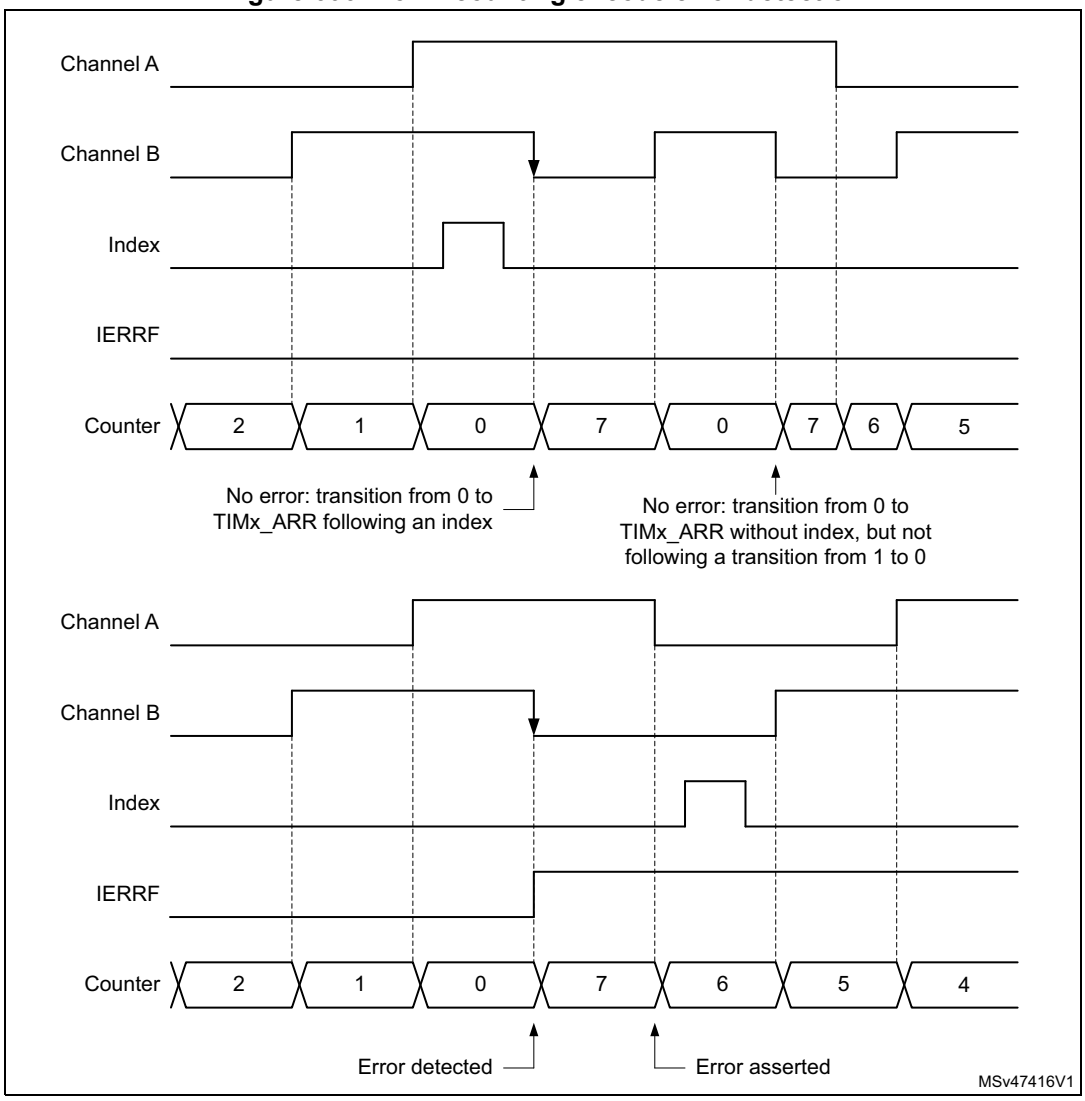
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 605*.

Figure 605. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 606*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 606. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

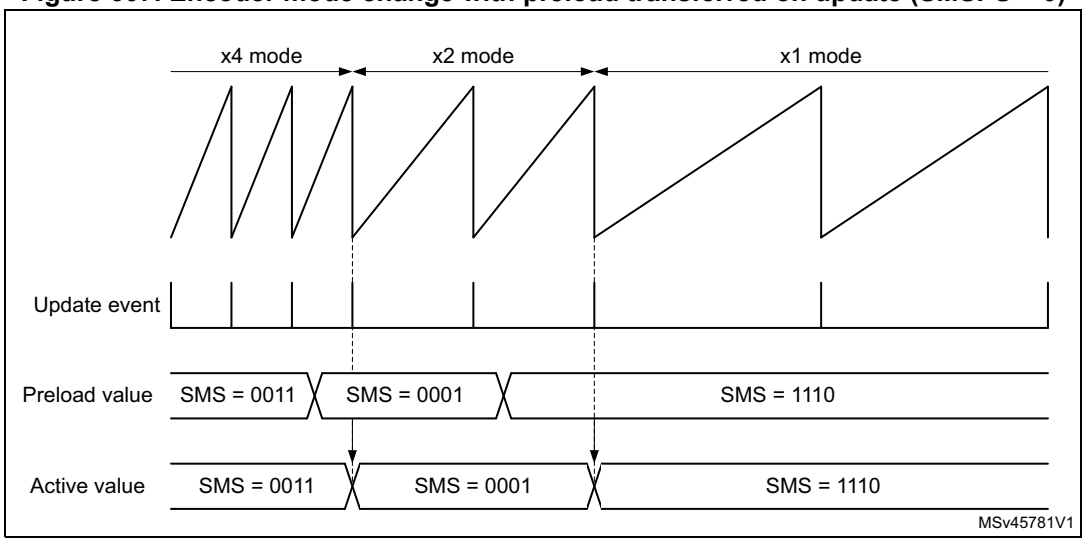
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 607*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 607. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

54.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

54.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

54.4.21 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 53.3.29: Interfacing with Hall sensors*.

54.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

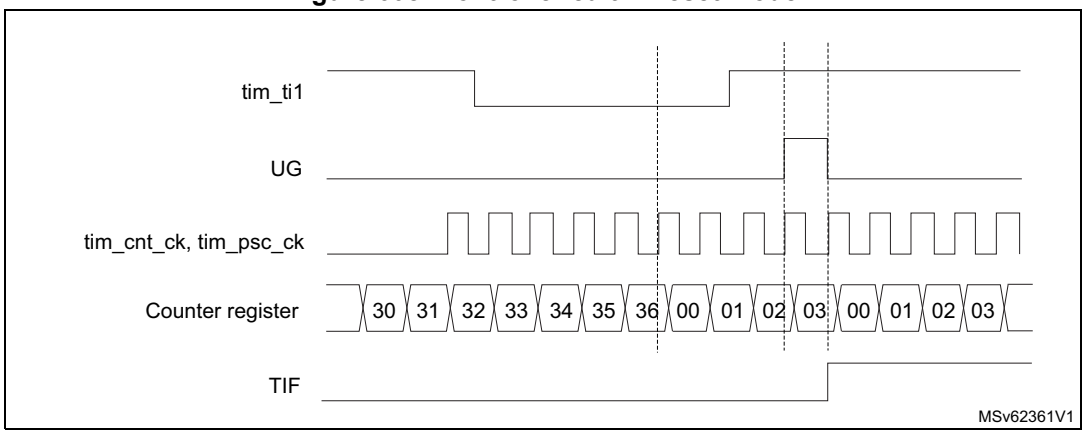
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the

trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 608. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

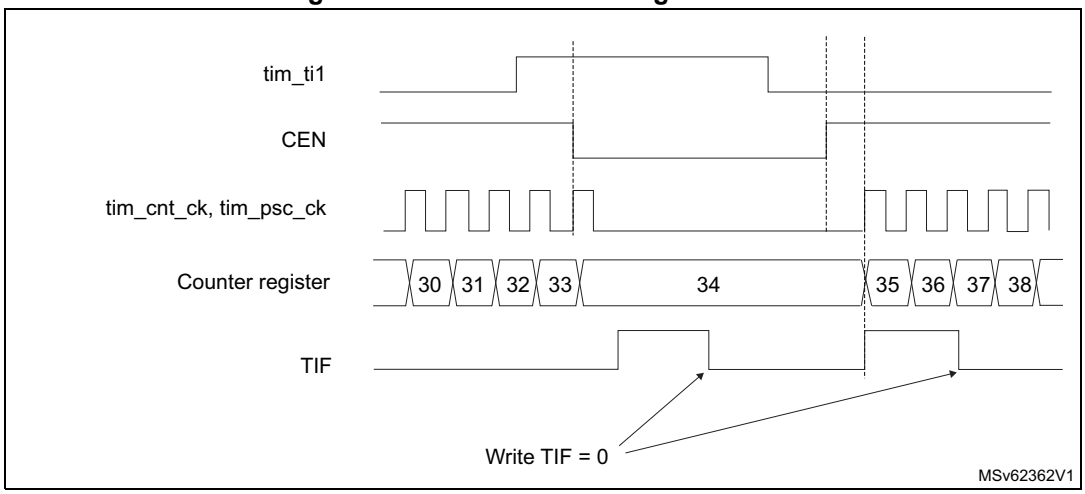
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 609. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

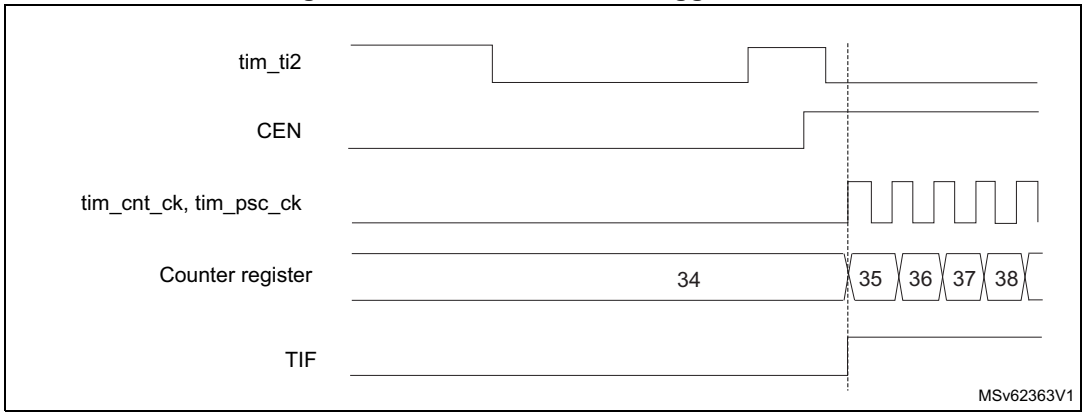
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 610. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

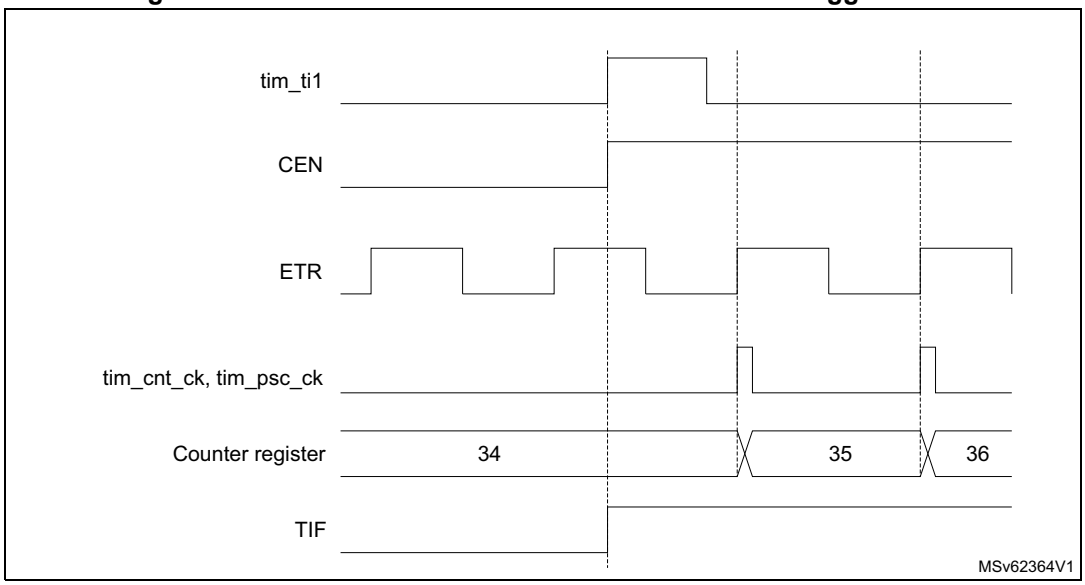
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 611. Control circuit in external clock mode 2 + trigger mode



54.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

Figure 612 and *Figure 613* show examples of master/slave timer connections.

Figure 612. Master/Slave timer example

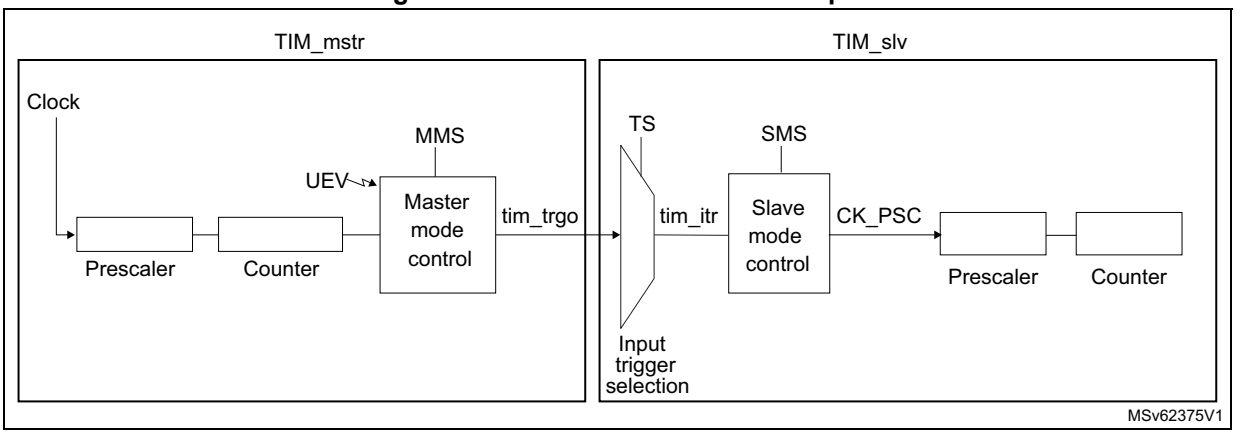
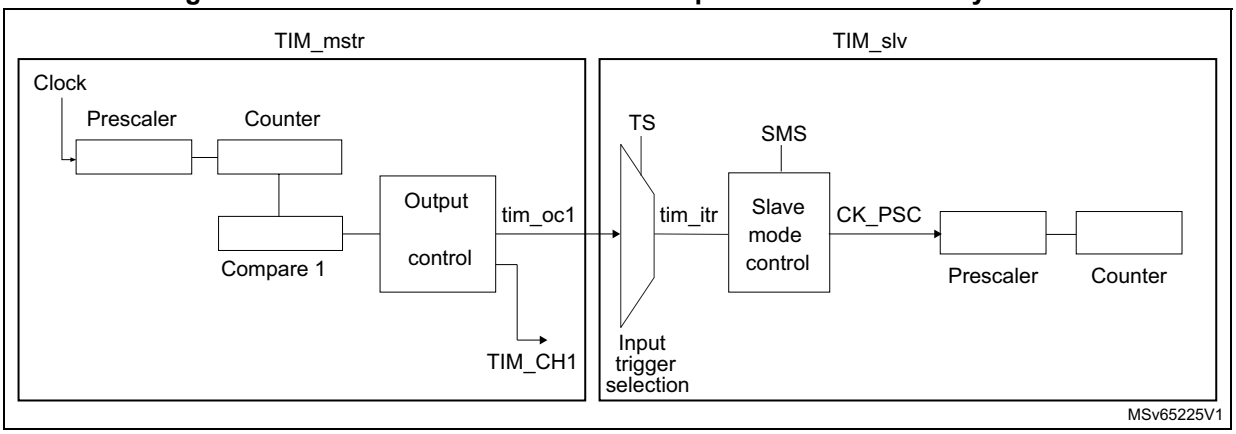


Figure 613. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 613) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 612*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

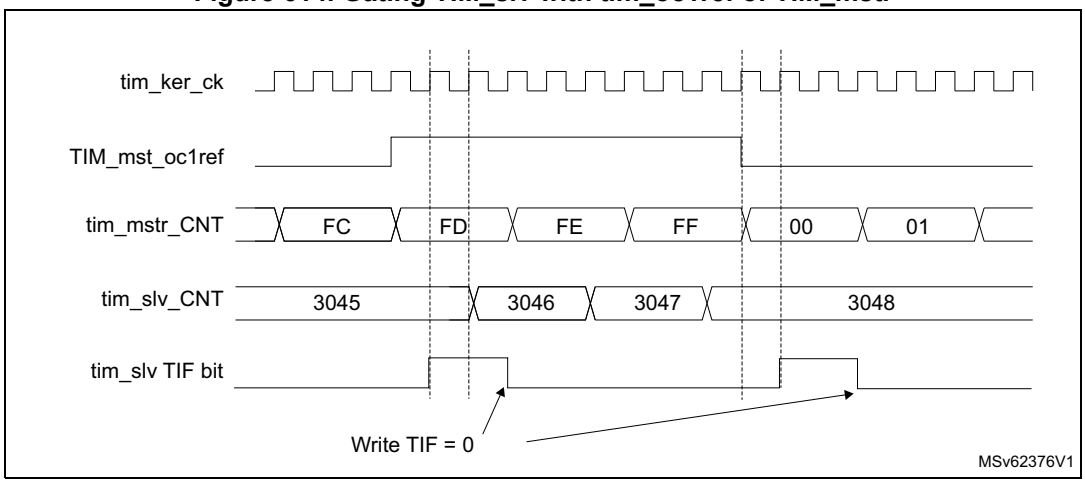
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 612* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 614. Gating TIM_slv with tim_oc1ref of TIM_mstr

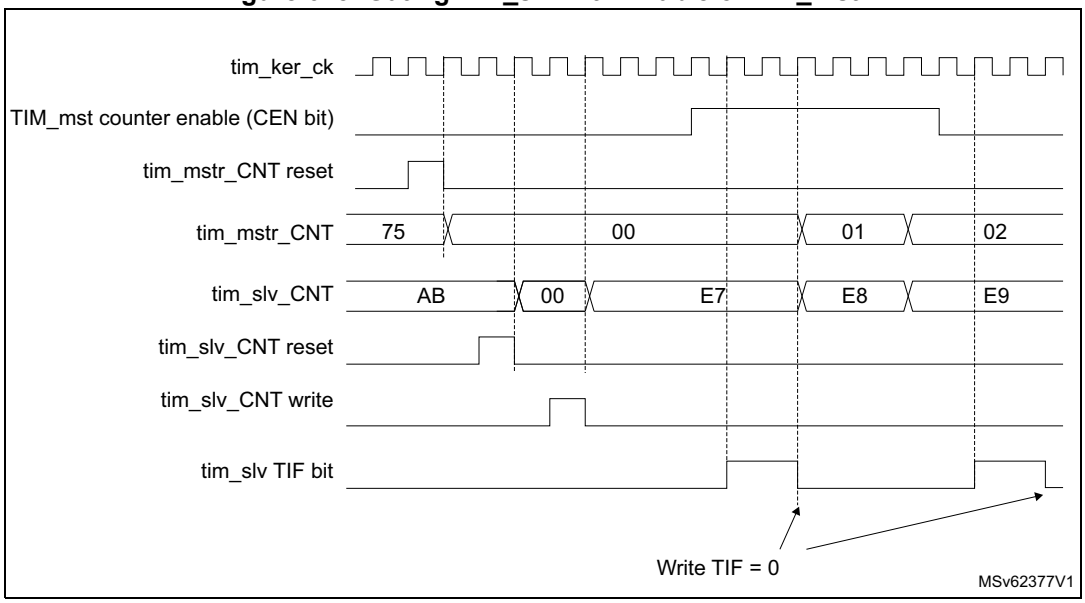


In the example in *Figure 614*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 615*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 615. Gating TIM_slv with Enable of TIM_mstr

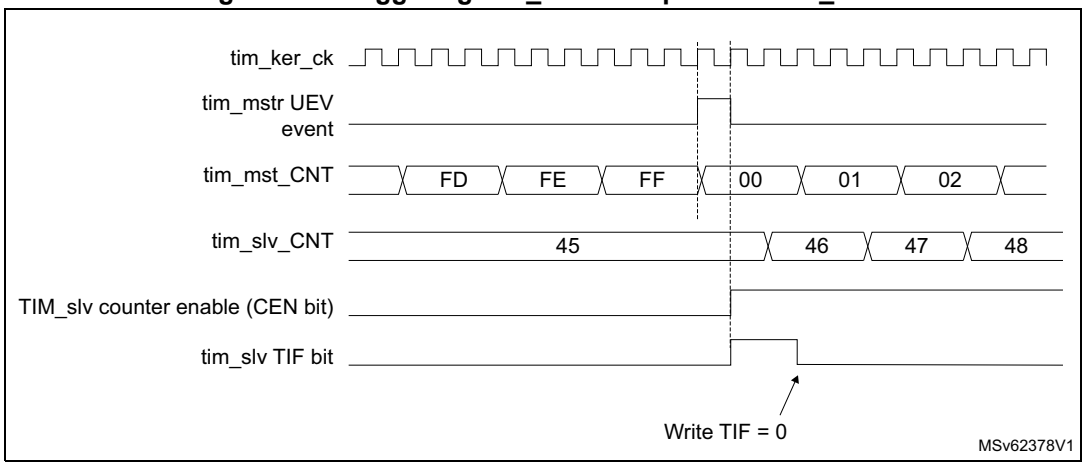


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 612* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

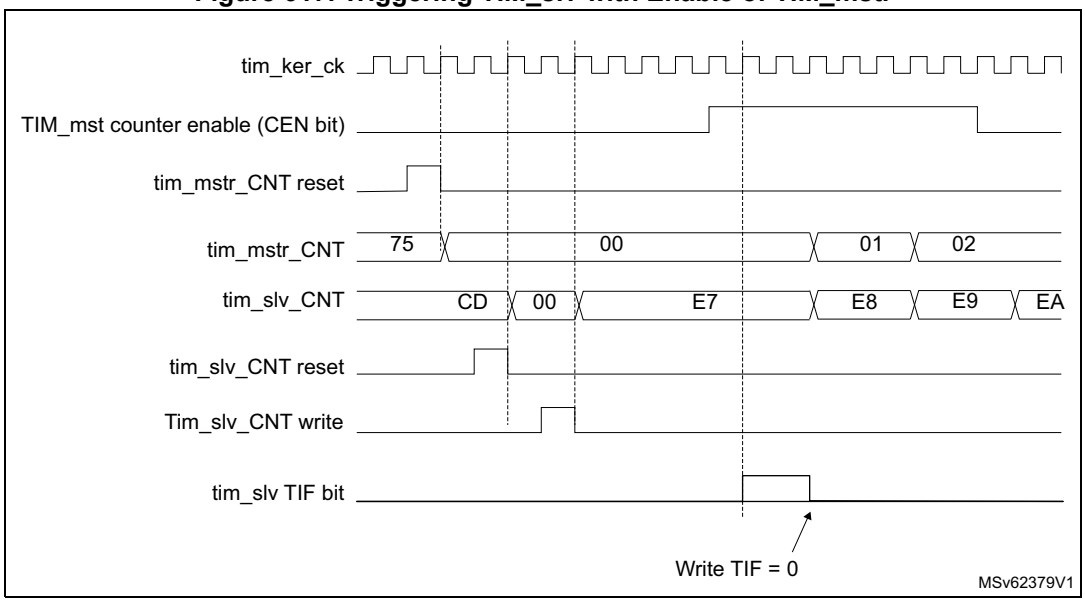
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 616. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 617* shows the behavior with the same configuration as in *Figure 616* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 617. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

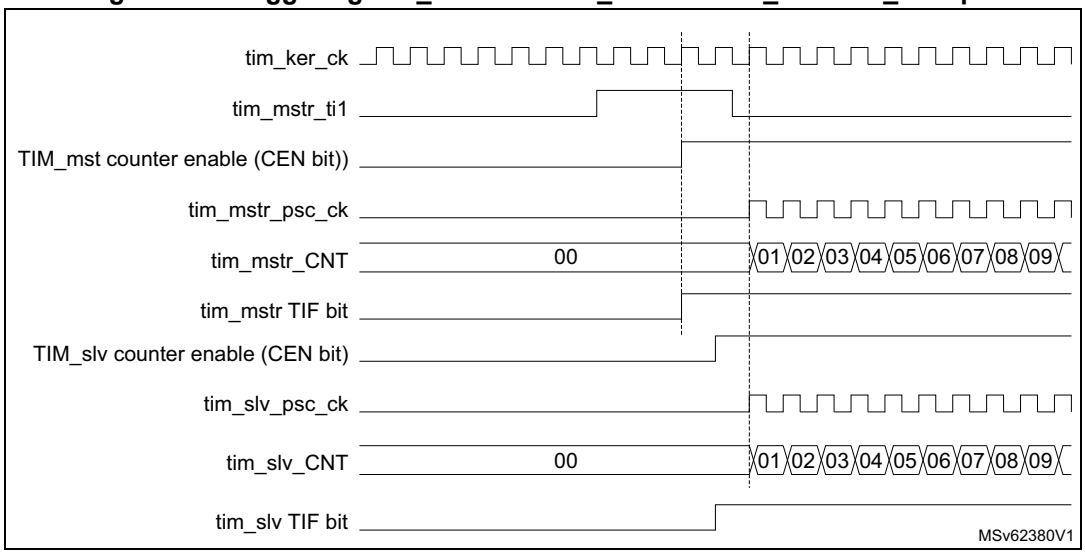
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 612* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 618. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.4.25 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 54.5.23: TIMx DMA control register (TIMx_DCR)(x = 2 to 5)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5,

and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

54.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 553*.

Table 553. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

54.4.27 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

54.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 554. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

54.4.29 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 555*.

Table 555. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

54.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

54.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

54.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 53.3.29: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

54.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional clock, x2 mode.

1101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

54.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

54.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

54.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

54.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

54.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

54.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMx_ECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

54.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 556. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

54.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

54.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

54.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 54.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

54.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

54.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

54.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

54.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

54.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

54.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

54.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

54.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

54.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

54.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

54.5.25 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 2 to 5)

Address offset: 0x3EC

Reset value: 0x0000 2011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width

0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width

0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear

0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

54.5.26 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 2 to 5)

Address offset: 0x3F0

Reset value: 0x2202 0104

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
- 0x0: Down-counter only
- 0x1: Up-counter only
- 0x2: Up/Down counter allowing center-aligned mode
- Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
- 0x0: DMA requests are not possible
- 0x1: Single DMA requests possible
- 0x2: All DMA features implemented
- Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
- 0x0: no break implemented
- 0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
- 0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
- 0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
- Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
- 0x0: External trigger circuit not implemented, tim_ocrefx cleared by tim_ocref_clr input
- 0x1: External trigger circuit implemented, tim_ocrefx cleared by tim_ocref_clr input
- 0x2: External trigger circuit implemented, tim_ocrefx cleared by tim_etrf signal
- 0x3: External trigger circuit implemented, tim_ocrefx cleared by tim_ocref_clr or tim_etrf (1 control bit “OCCS” added in GPT_SMCR)
- Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
- 0x0: repetition counter not implemented
- 0x1: 8-bit repetition counter implemented
- 0x2: 16-bit repetition counter implemented
- Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

54.5.27 TIMx IP version register (TIMx_VERR)(x = 2 to 5)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

54.5.28 TIMx IP identification register (TIMx_IPIDR)(x = 2 to 5)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

54.5.29 TIMx size ID register (TIMx_SIDR)(x = 2 to 5)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

54.5.30 TIMx register map

TIMx registers are mapped as described in the table below.

Table 557. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value							0																		0	0	0	0	0			

Table 557. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value								0	0	0	0							0		0	0	0	0	0		0		0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	TIMx_CNT	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR (x = 2 to 5)	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved																																
0x034	TIMx_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 557. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	TIMx_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	TIMx_HWCFGR 2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_ WIDTH16[3:0]				PSC_ LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIMx_HWCFGR 1	CNT_ DIR[3:0]				DMA_ IMPL[3:0]				BK_ IMPL[3:0]				ETR_ IMPL[3:0]				RCR_ IMPL[3:0]				NB_OF_ SM[3:0]				NB_OF_ DT[3:0]				NB_OF_ CC[3:0]			
	Reset value	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0
0x3F4	TIMx_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIMx_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIMx_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

55 Basic timers (TIM6/TIM7)

55.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

55.2 TIM6/TIM7 main features

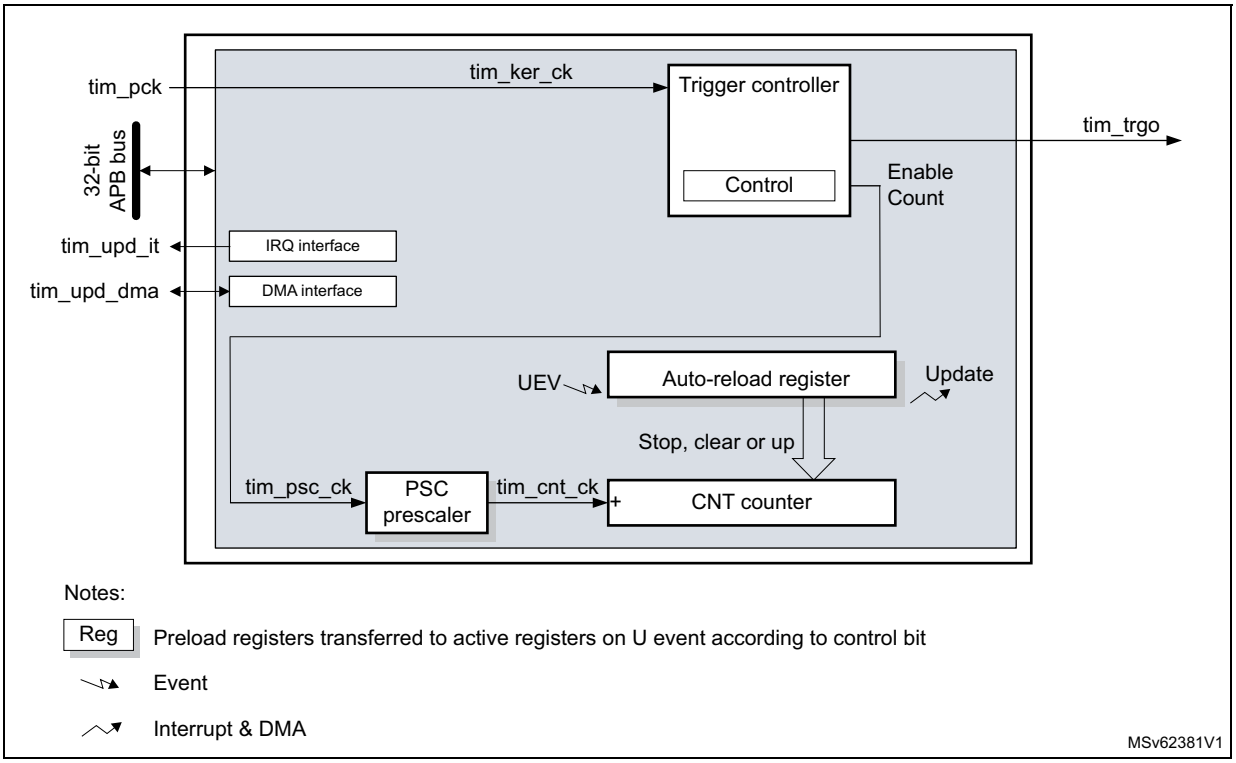
Basic timer (TIM6/TIM7) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.

55.3 TIM6/TIM7 functional description

55.3.1 TIM6/TIM7 block diagram

Figure 619. Basic timer block diagram



55.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 558. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

55.3.3 TIM6/TIM7 clocks

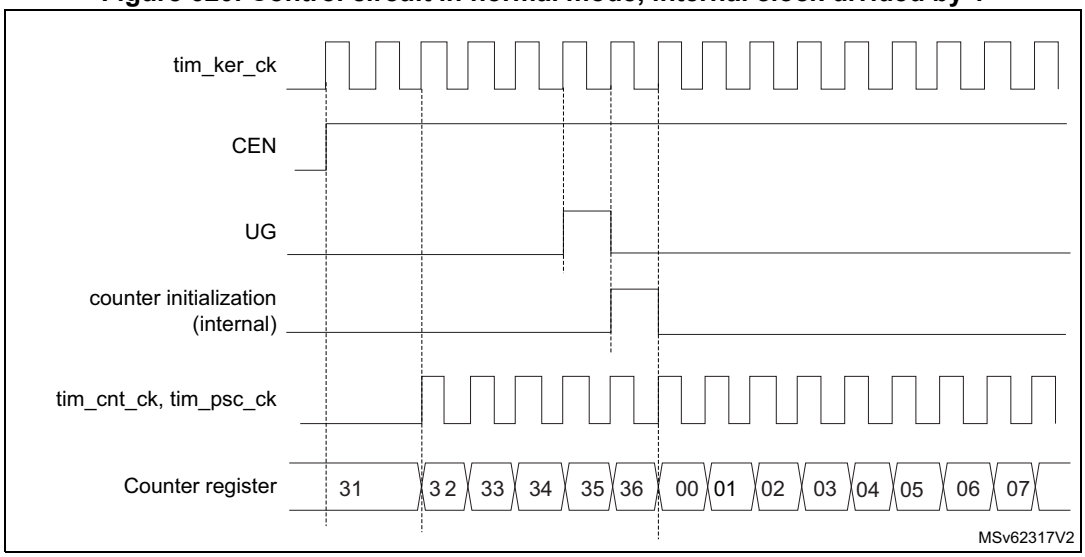
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 620 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 620. Control circuit in normal mode, internal clock divided by 1



55.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 621 and *Figure 622* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 621. Counter timing diagram with prescaler division change from 1 to 2

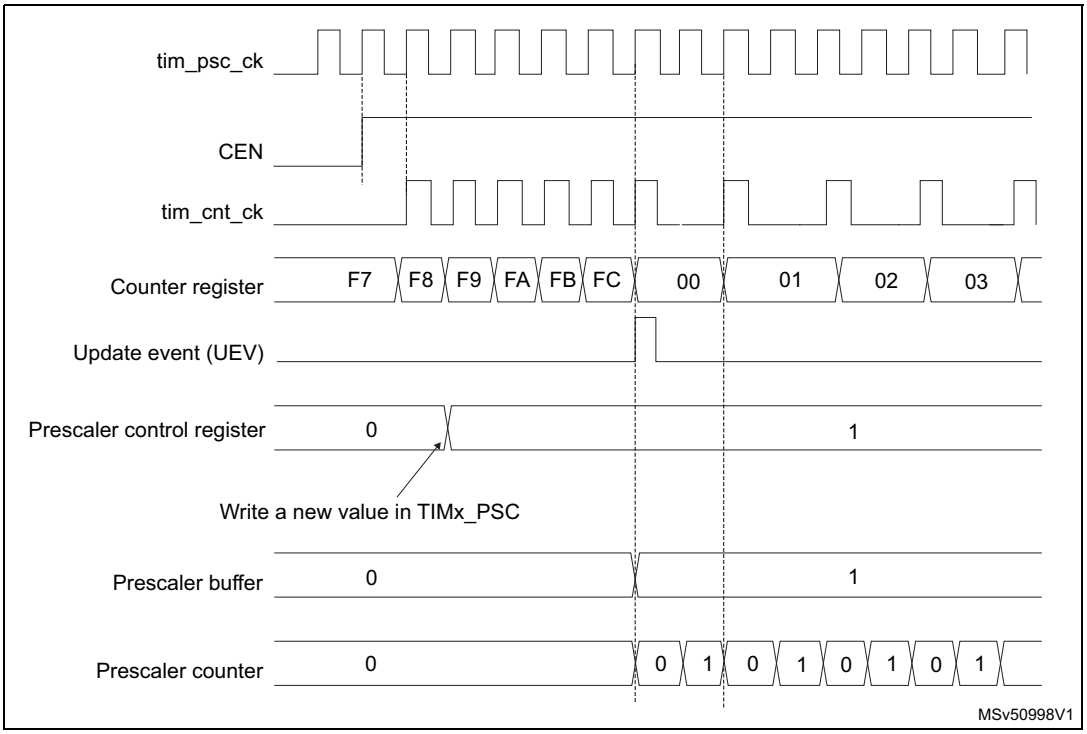
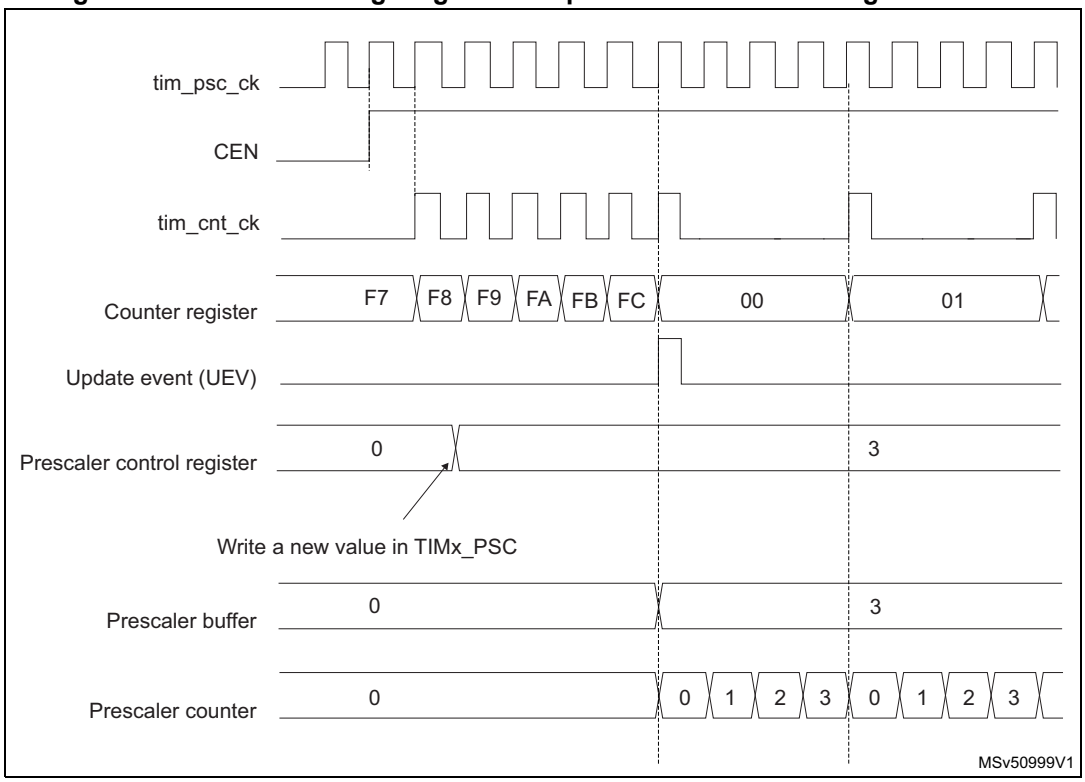


Figure 622. Counter timing diagram with prescaler division change from 1 to 4



55.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 623. Counter timing diagram, internal clock divided by 1

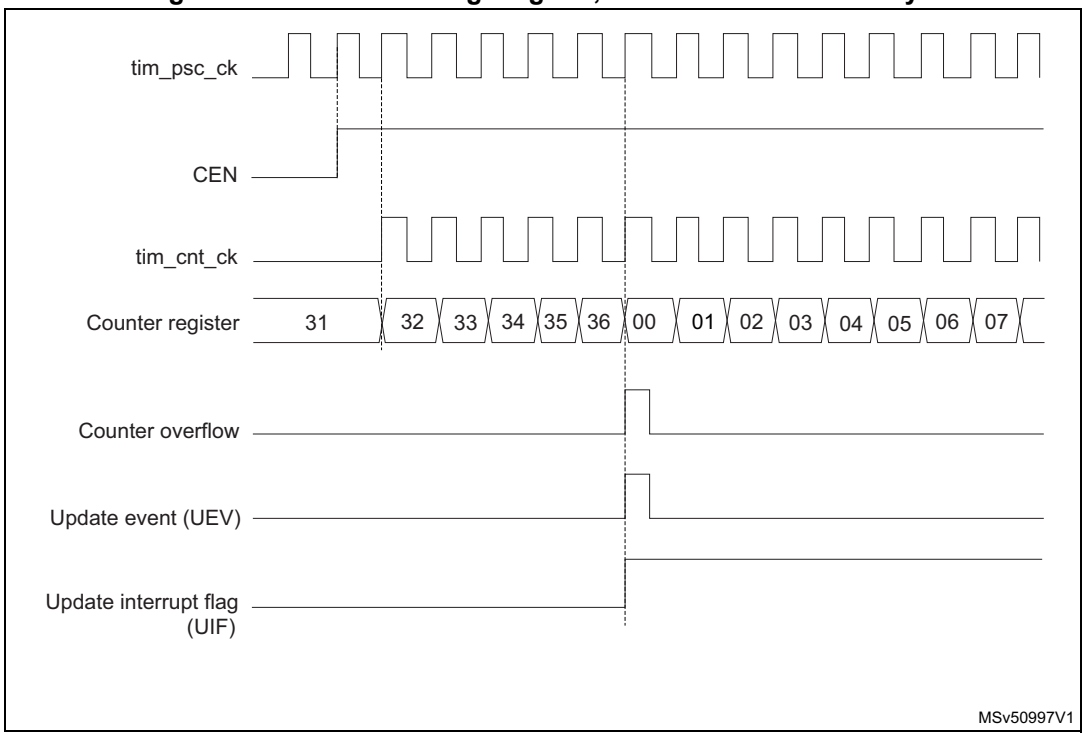


Figure 624. Counter timing diagram, internal clock divided by 2

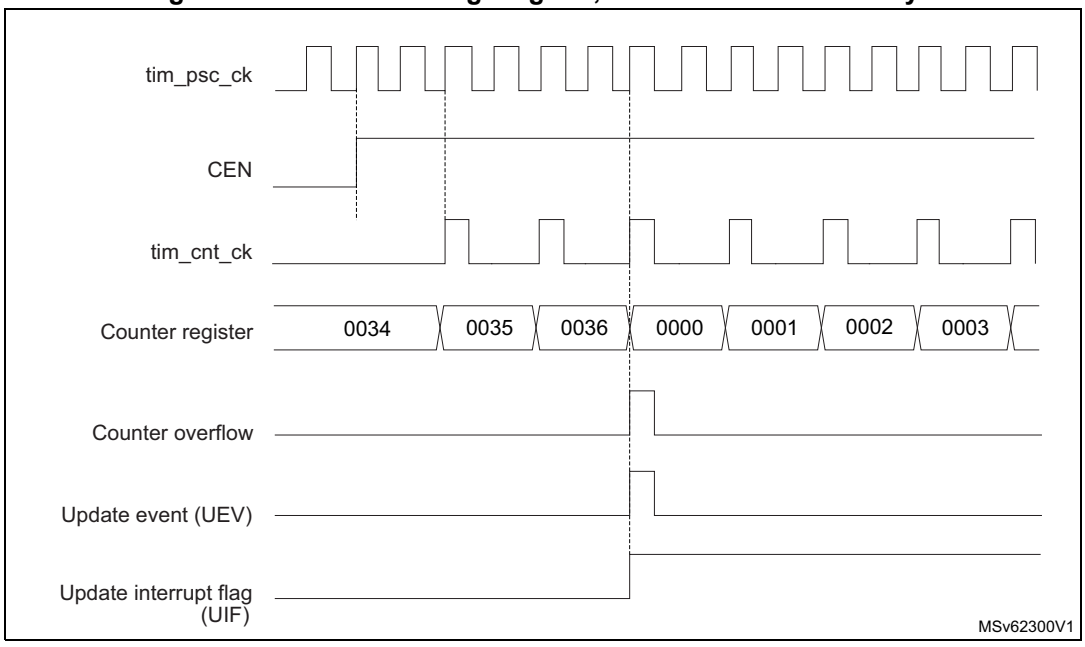


Figure 625. Counter timing diagram, internal clock divided by 4

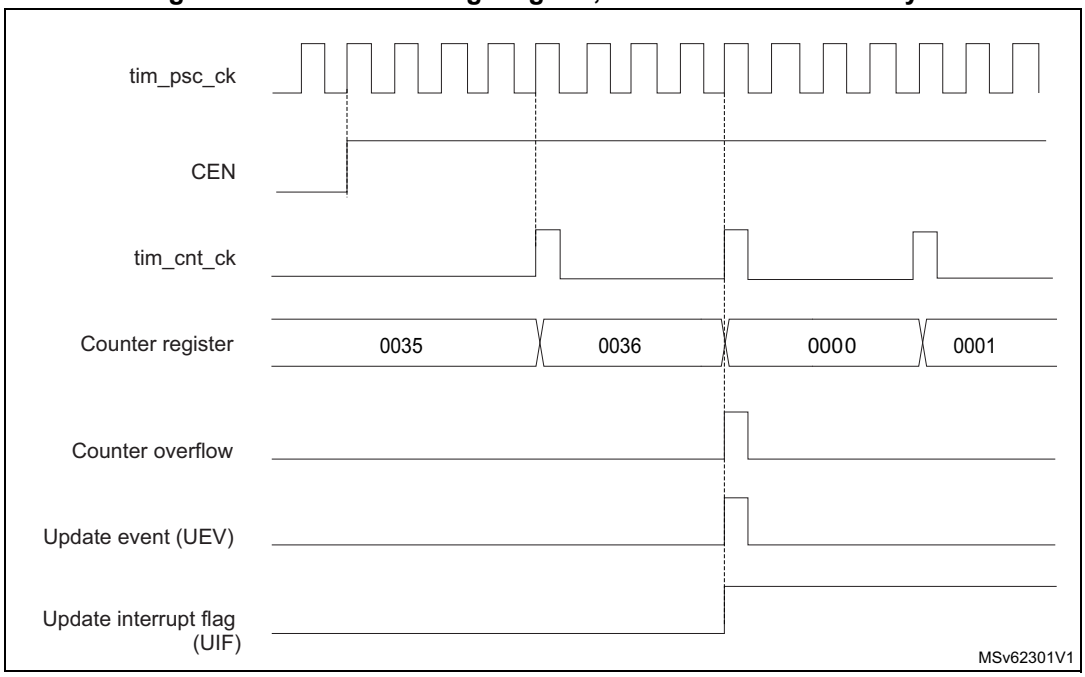


Figure 626. Counter timing diagram, internal clock divided by N

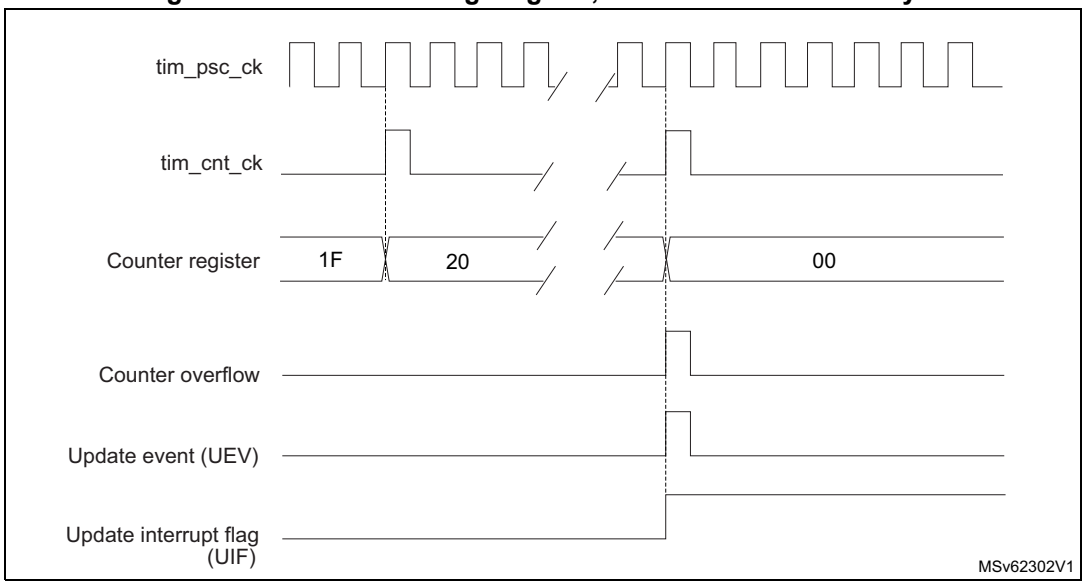


Figure 627. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

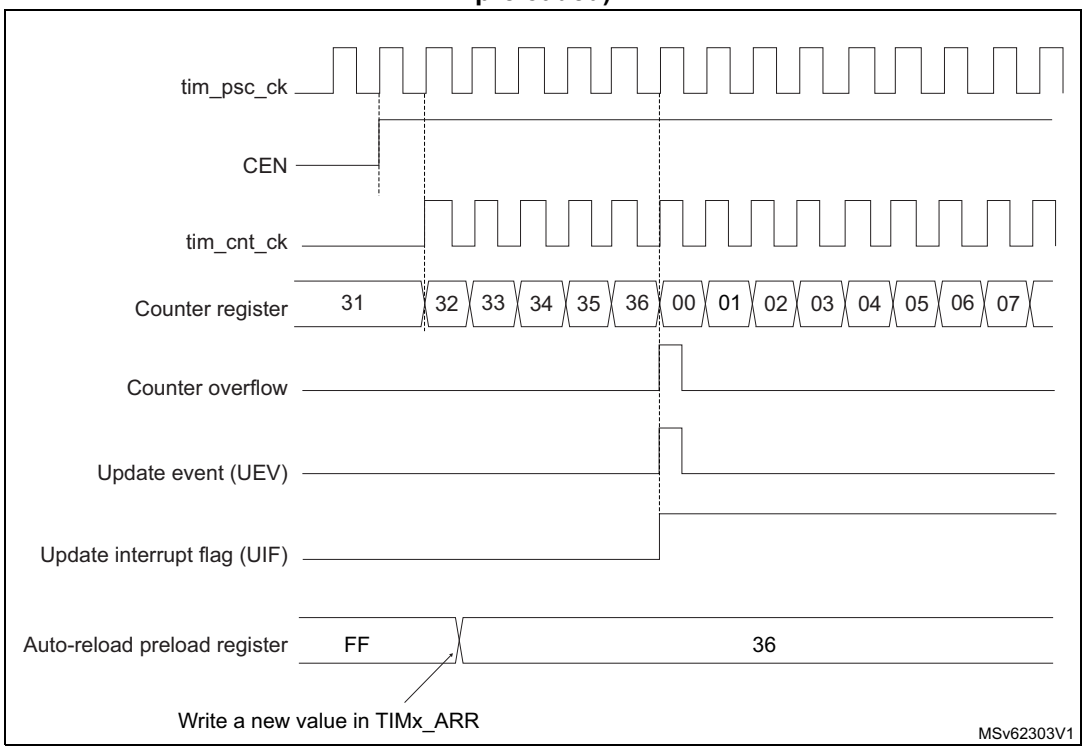
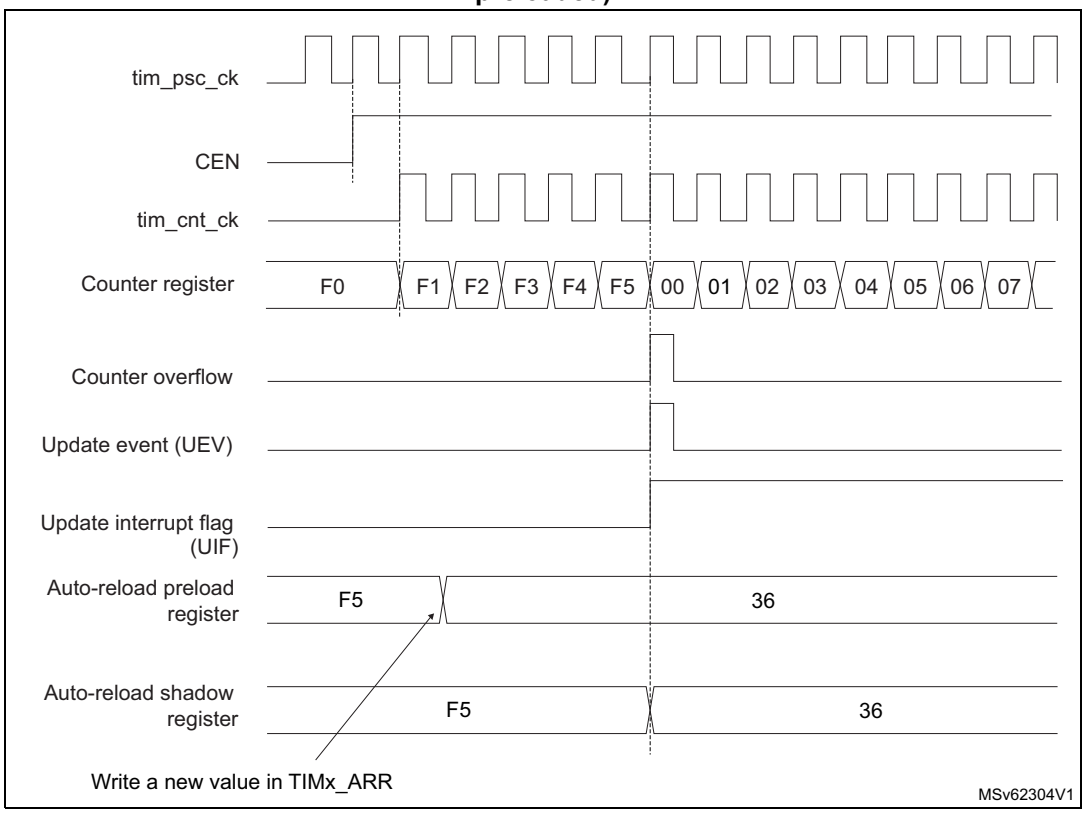


Figure 628. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



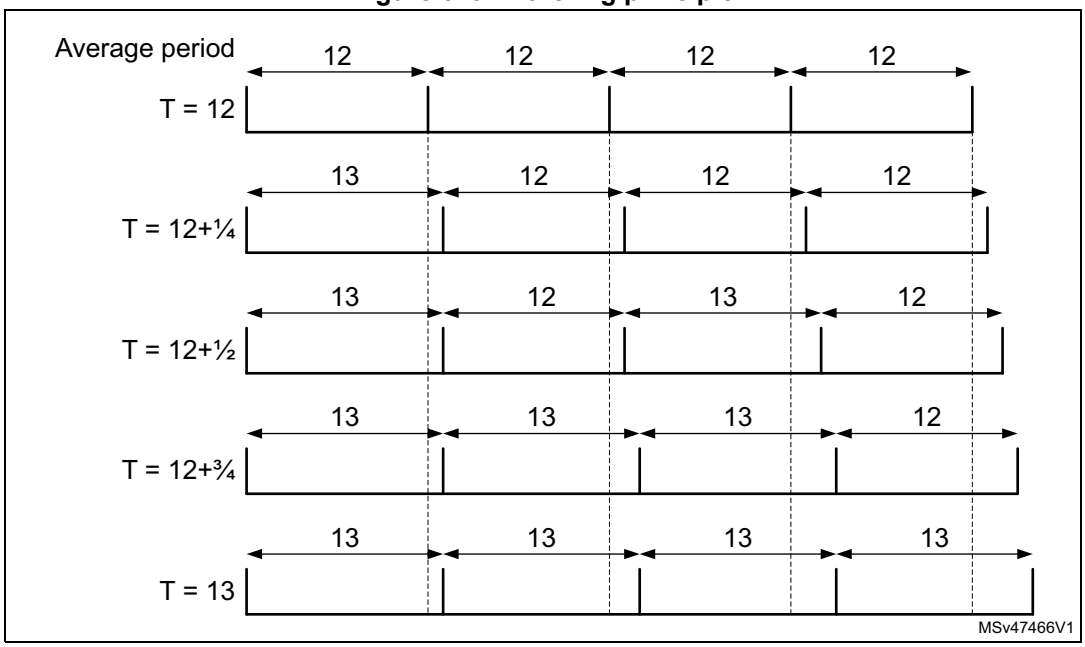
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 629 presents the dithering principle applied to four consecutive counting periods.

Figure 629. Dithering principle

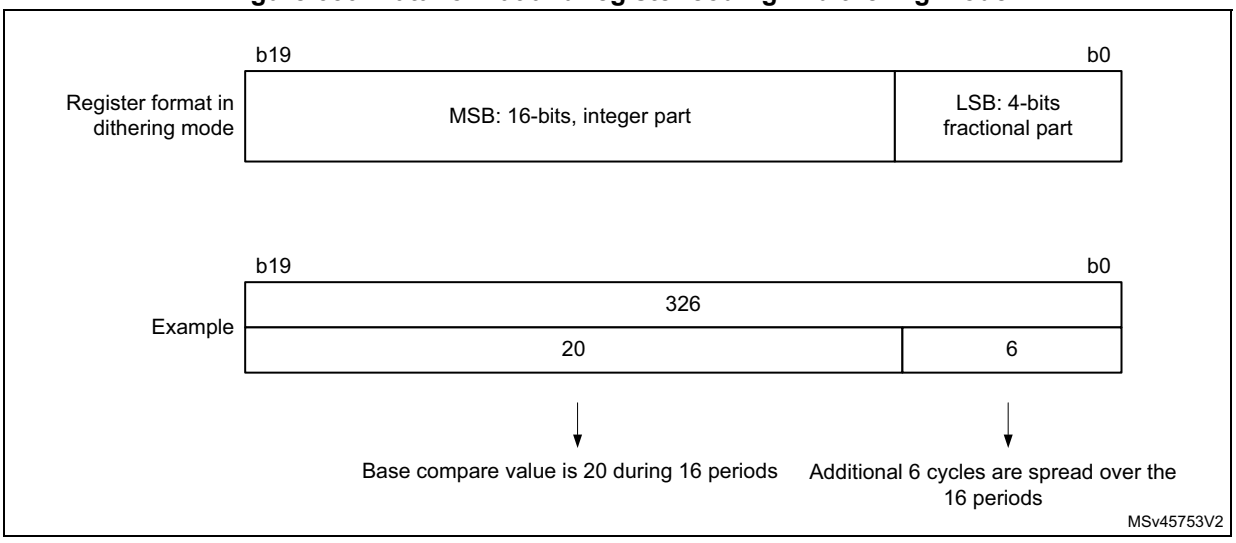


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 630* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 630. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

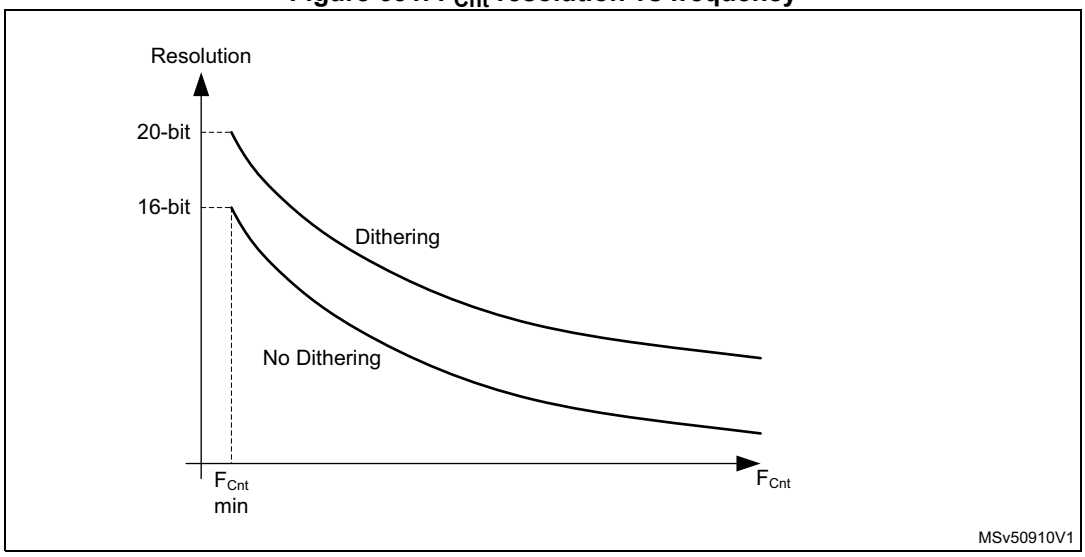
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

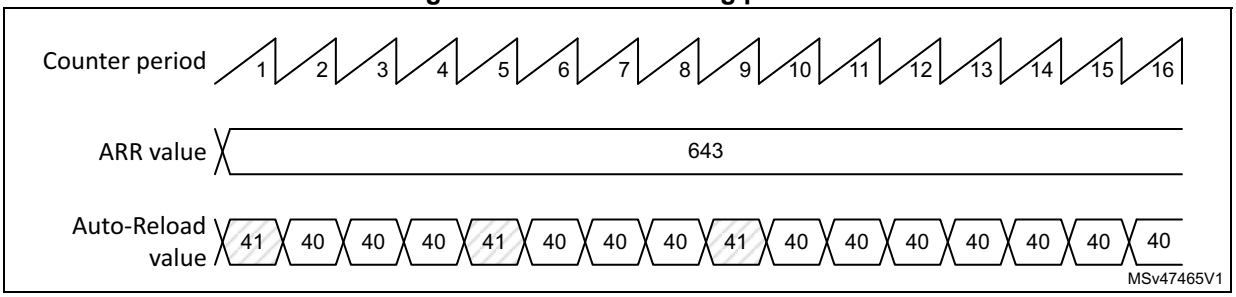
As shown on *Figure 631*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 631. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 632*.

Figure 632. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 559*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 559. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

55.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

55.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

55.3.8 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 560*.

Table 560. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

55.3.9 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

55.3.10 TIM6/TIM7 low-power modes

Table 561. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

55.3.11 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 562*.

Table 562. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

55.4 TIM6/TIM7 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

55.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

55.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

55.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

55.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

55.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

55.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

55.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

55.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 55.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

55.4.9 TIMx hardware configuration 2 register (TIMx_HWCFGR2)(x = 6 to 7)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width
0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width
0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear
0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

55.4.10 TIMx hardware configuration 1 register (TIMx_HWCFGR1)(x = 6 to 7)

Address offset: 0x3F0

Reset value: 0x1100 0100

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
0x0: Down-counter only
0x1: Up-counter only
0x2: Up/Down counter allowing center-aligned mode
Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
0x0: DMA requests are not possible
0x1: Single DMA requests possible
0x2: All DMA features implemented
Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
0x0: no break implemented
0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
0x0:External trigger circuit not implemented, tim_ocrefx cleared by tim_ocref_clr input
0x1:External trigger circuit implemented, tim_ocrefx cleared by tim_ocref_clr input
0x2:External trigger circuit implemented, tim_ocrefx cleared by tim_etrf signal
0x3:External trigger circuit implemented, tim_ocrefx cleared by tim_ocref_clr or tim_etrf (1 control bit “OCCS” added in GPT_SMCR)
Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
0x0: repetition counter not implemented
0x1: 8-bit repetition counter implemented
0x2: 16-bit repetition counter implemented
Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

55.4.11 TIMx IP version register (TIMx_VERR)(x = 6 to 7)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

55.4.12 TIMx IP identification register (TIMx_IPIDR)(x = 6 to 7)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

55.4.13 TIMx size ID register (TIMx_SIDR)(x = 6 to 7)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

55.4.14 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 563. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3EC	TIMx_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_WIDTH16 [3:0]				PSC_LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIMx_HWCFGR1	CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]				RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
	Reset value	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x3F4	TIMx_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	1	0	0	0	0	1	1

Table 563. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F8	TIMx_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIMx_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

56 General-purpose timers (TIM10/TIM11/TIM12/TIM13/TIM14)

56.1 TIM10/TIM11/TIM12/TIM13/TIM14 introduction

The TIM10/TIM11/TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM10/TIM11/TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 56.4.20: Timer synchronization (TIM12 only)*.

56.2 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare

56.3 TIM10/TIM11/TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

56.4 TIM10/TIM11/TIM12/TIM13/TIM14 functional description

56.4.1 Block diagram

Figure 633. General-purpose timer block diagram (TIM12)

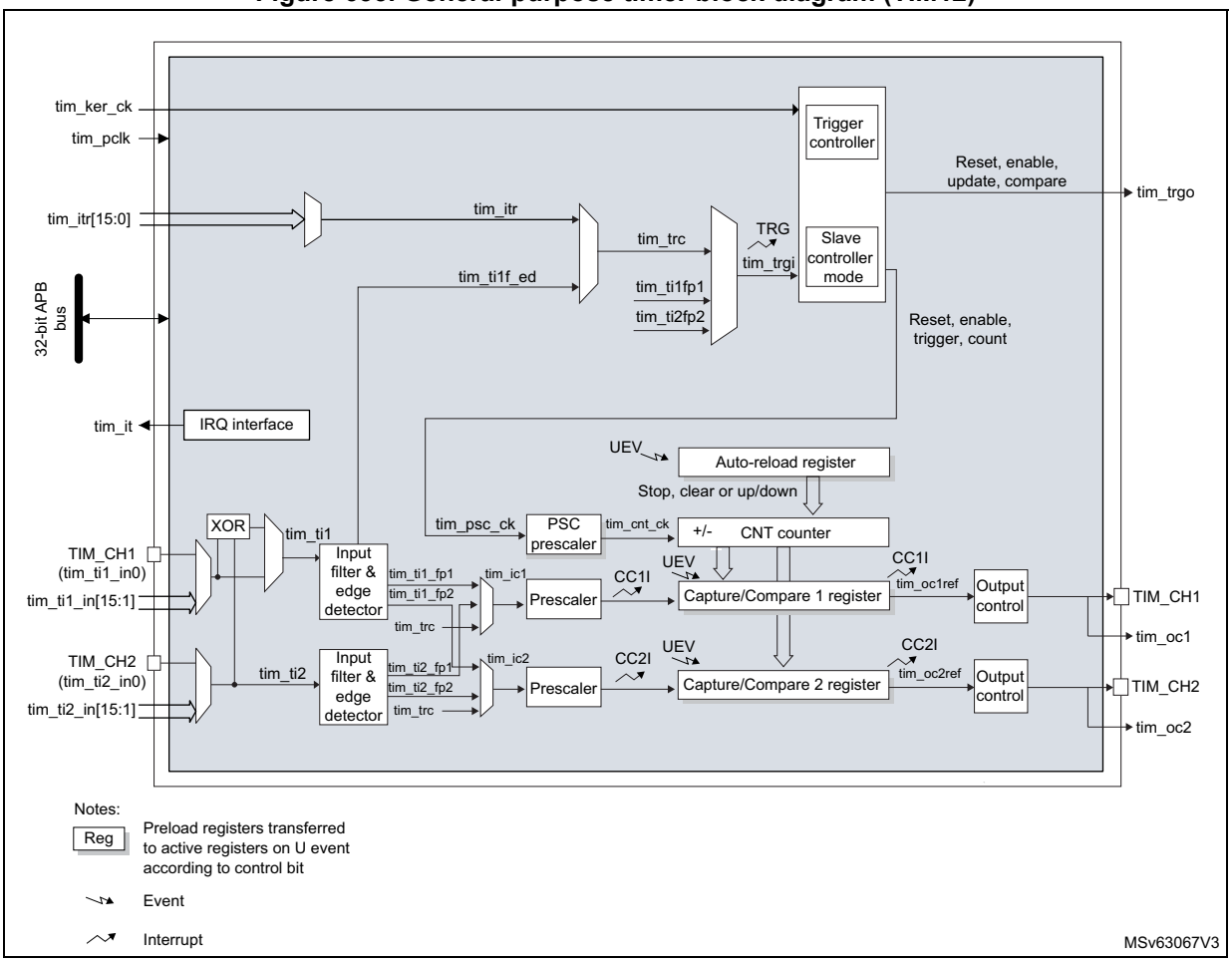
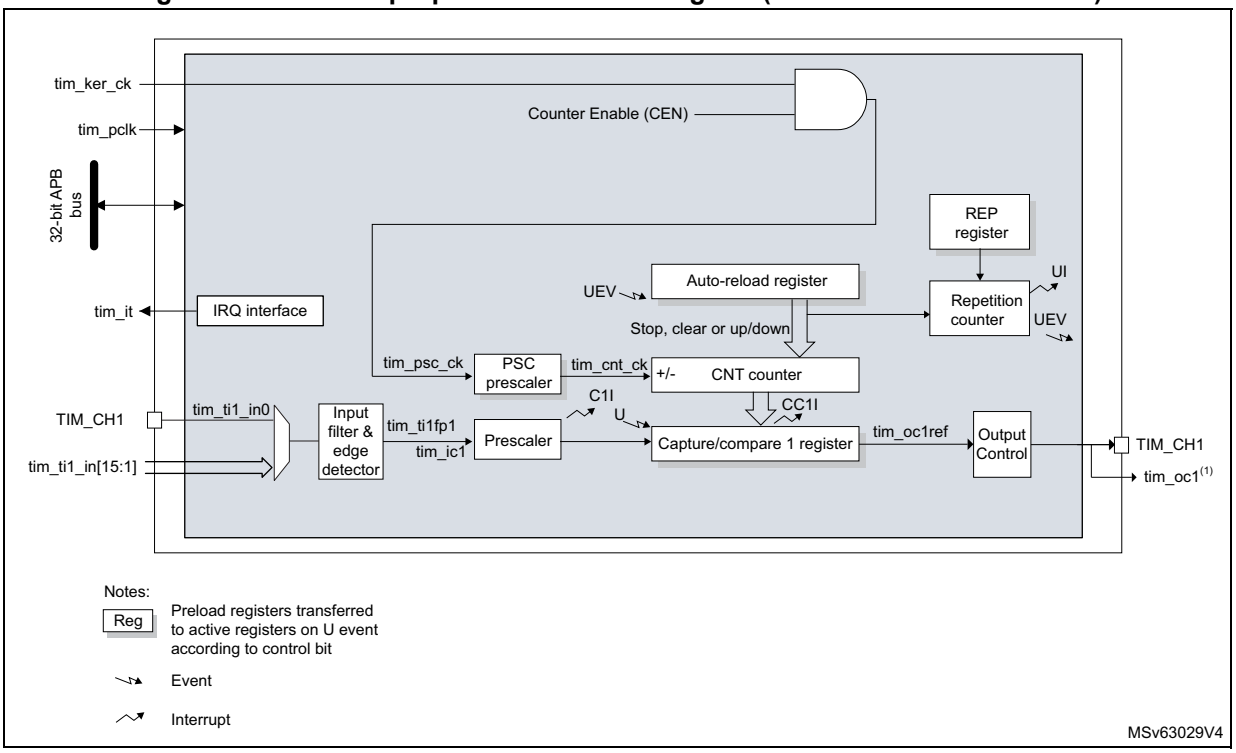


Figure 634. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14)



1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 56.4.21: Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)* for details.

56.4.2 TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals

Table 564 and *Table 565* in this section summarize the TIM inputs and outputs.

Table 564. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

1. Available for TIM12 only.

Table 565. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_oc1 tim_oc2(1)	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).

Table 565. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests

1. Available for TIM12 only.

Table 566 and *Table 567* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 566. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources				
	TIM10	TIM11	TIM12	TIM13	TIM14
tim_ti1_in0	TIM10_CH1	TIM11_CH1	TIM12_CH1	TIM13_CH1	TIM14_CH1
tim_ti1_in1	Reserved	Reserved	MSI/256	Reserved	Reserved
tim_ti1_in2			HSI/1024		
tim_ti1_in[15:3]	Reserved				

Table 567. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM12
tim_ti2_in0	TIM12_CH2
tim_ti2_in[15:1]	Reserved

Table 568 lists the internal sources connected to the tim_itr input multiplexer.

Table 568. TIMx internal trigger connection

TIMx	TIM12
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	Reserved

Table 568. TIMx internal trigger connection (continued)

TIMx	TIM12
tim_itr7	tim13_oc1
tim_itr8	tim14_oc1
tim_itr9	tim15_trgo
tim_itr10	tim16_oc1
tim_itr11	tim17_oc1
tim_itr12	Reserved
tim_itr13	tim10_oc1
tim_itr14	tim11_oc1
tim_itr15	Reserved

56.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 635 and *Figure 636* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 635. Counter timing diagram with prescaler division change from 1 to 2

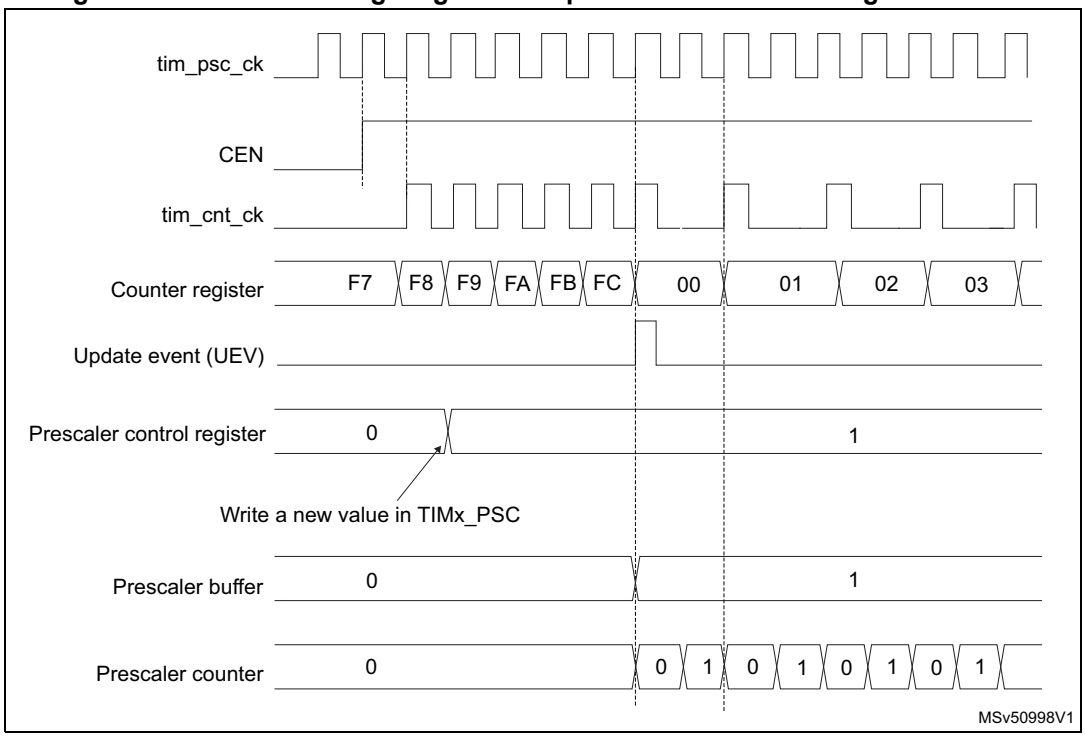
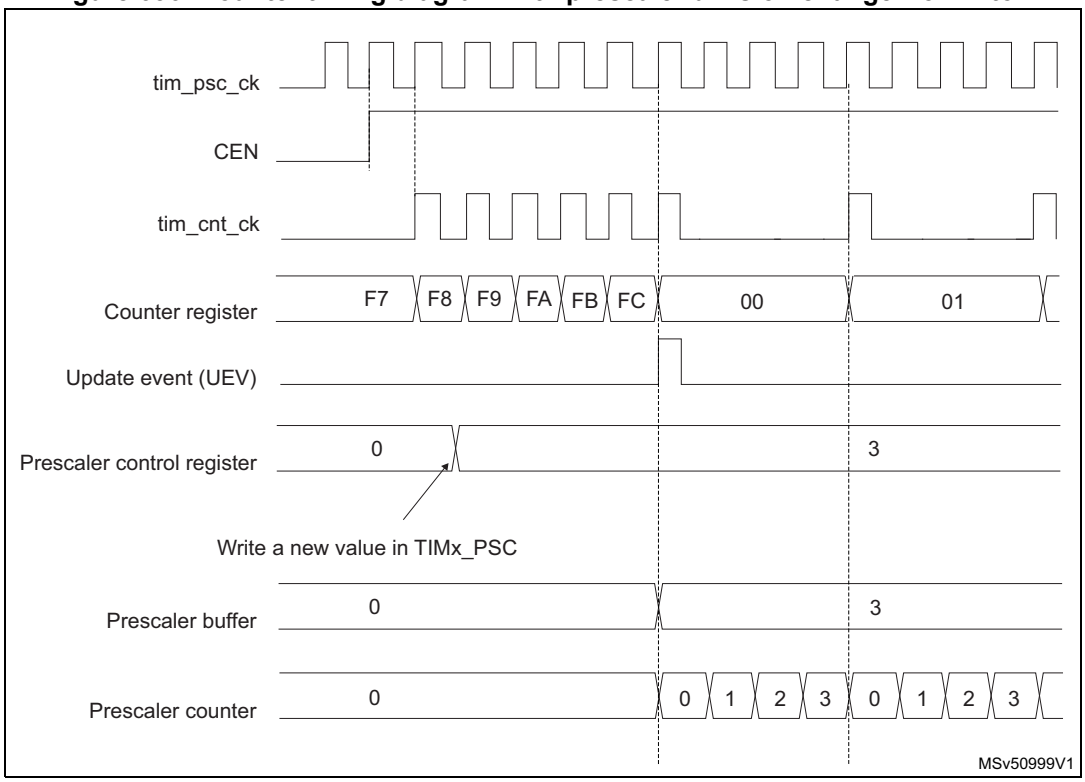


Figure 636. Counter timing diagram with prescaler division change from 1 to 4



56.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 637. Counter timing diagram, internal clock divided by 1

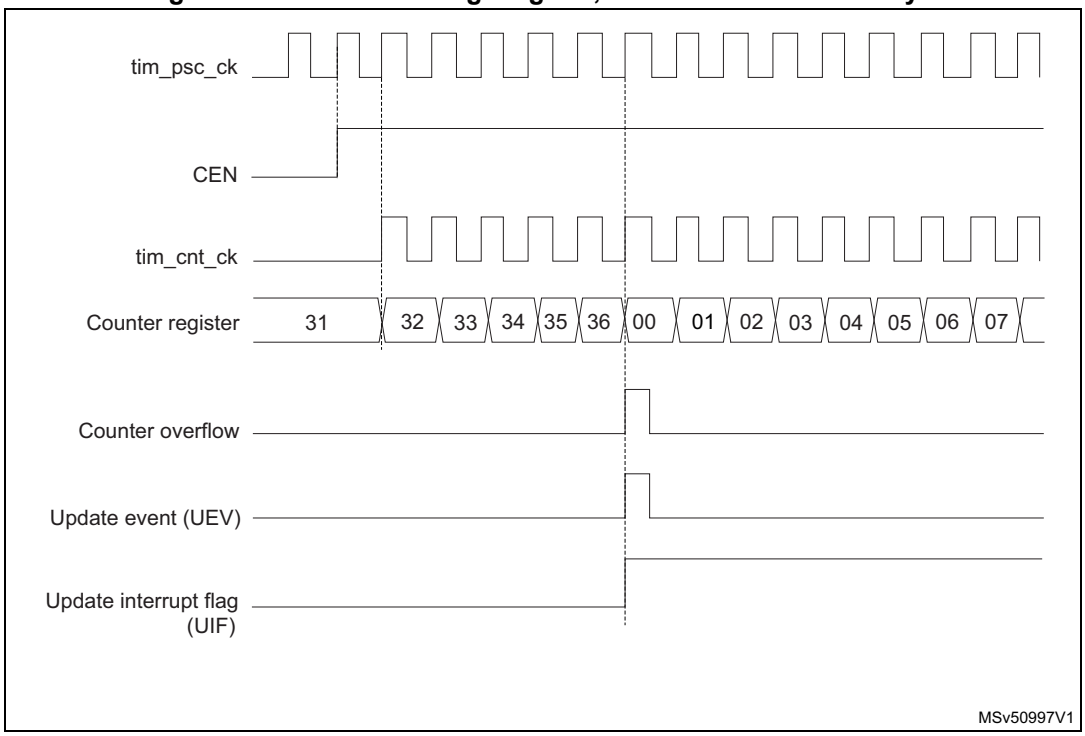


Figure 638. Counter timing diagram, internal clock divided by 2

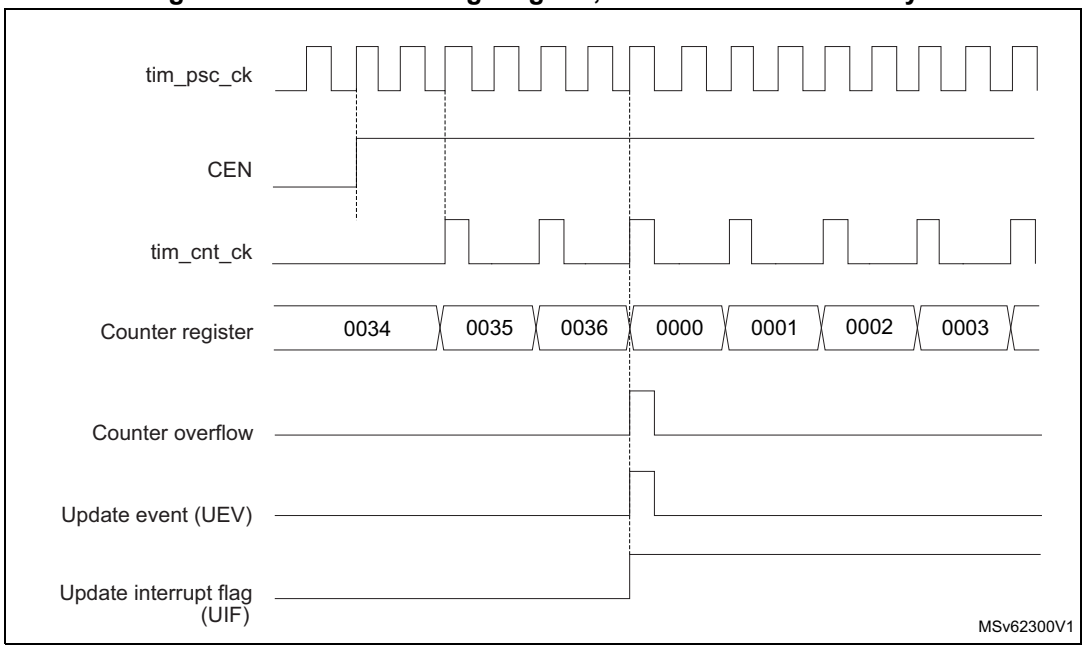


Figure 639. Counter timing diagram, internal clock divided by 4

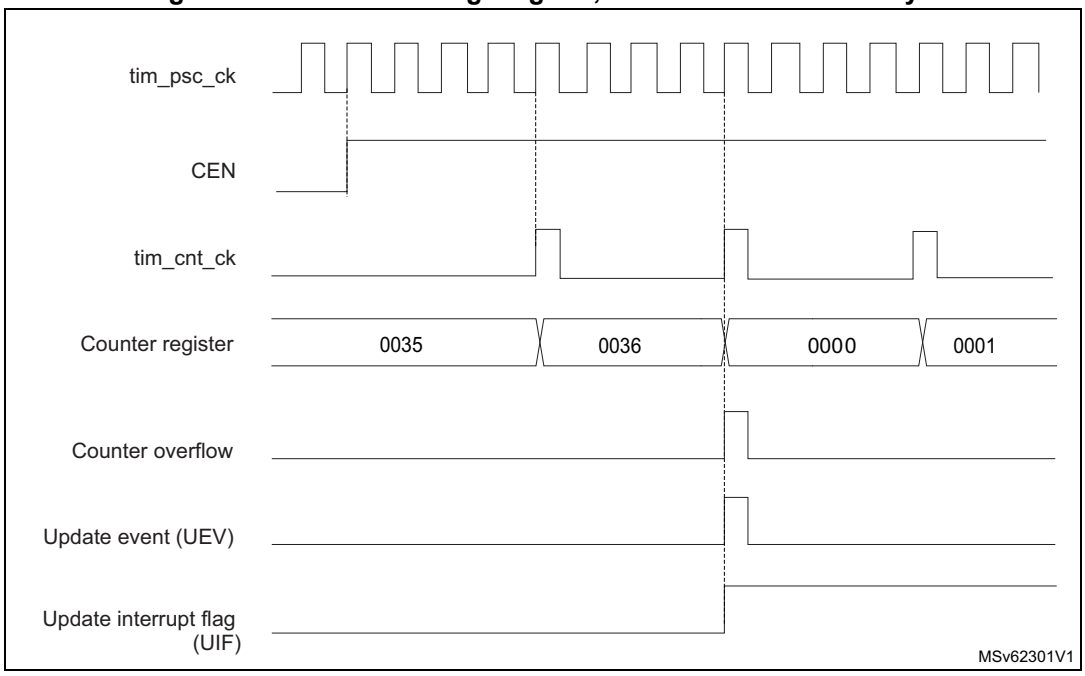


Figure 640. Counter timing diagram, internal clock divided by N

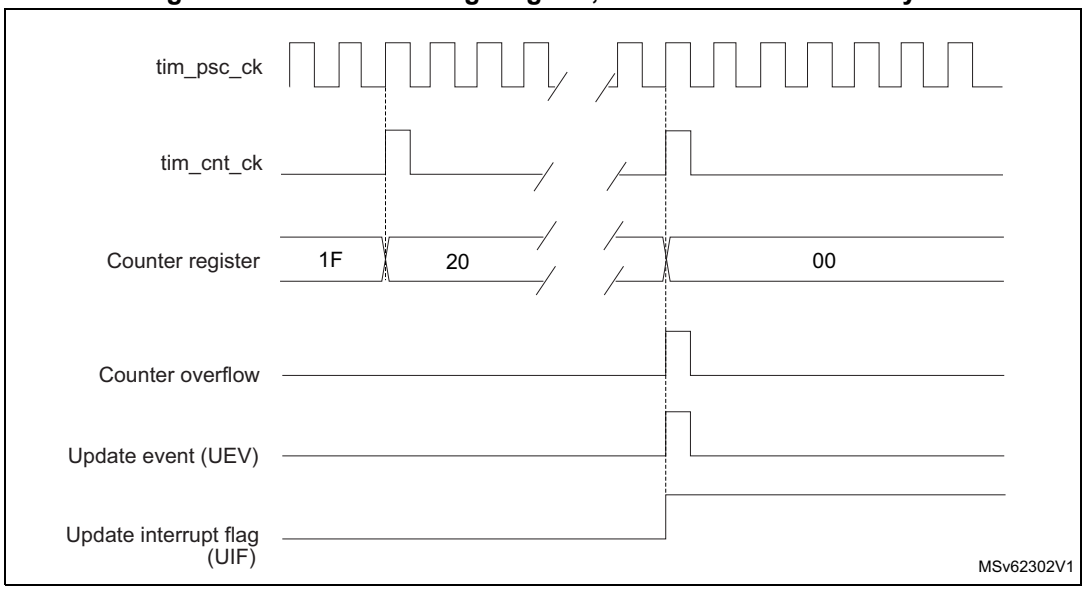


Figure 641. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

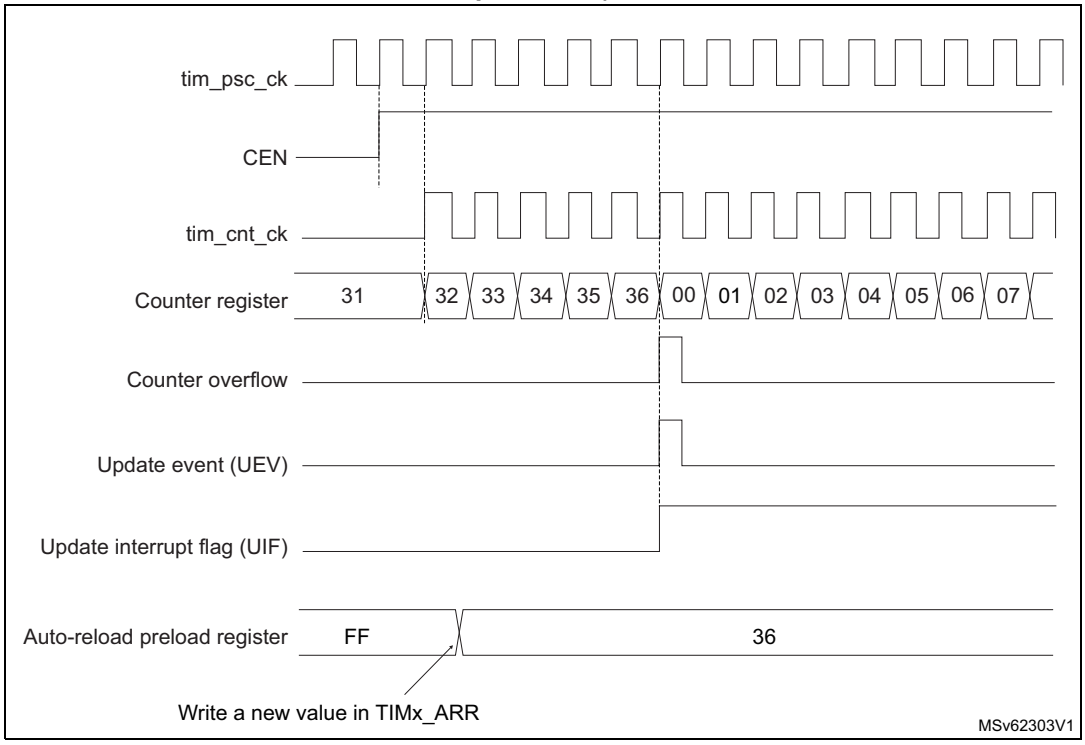
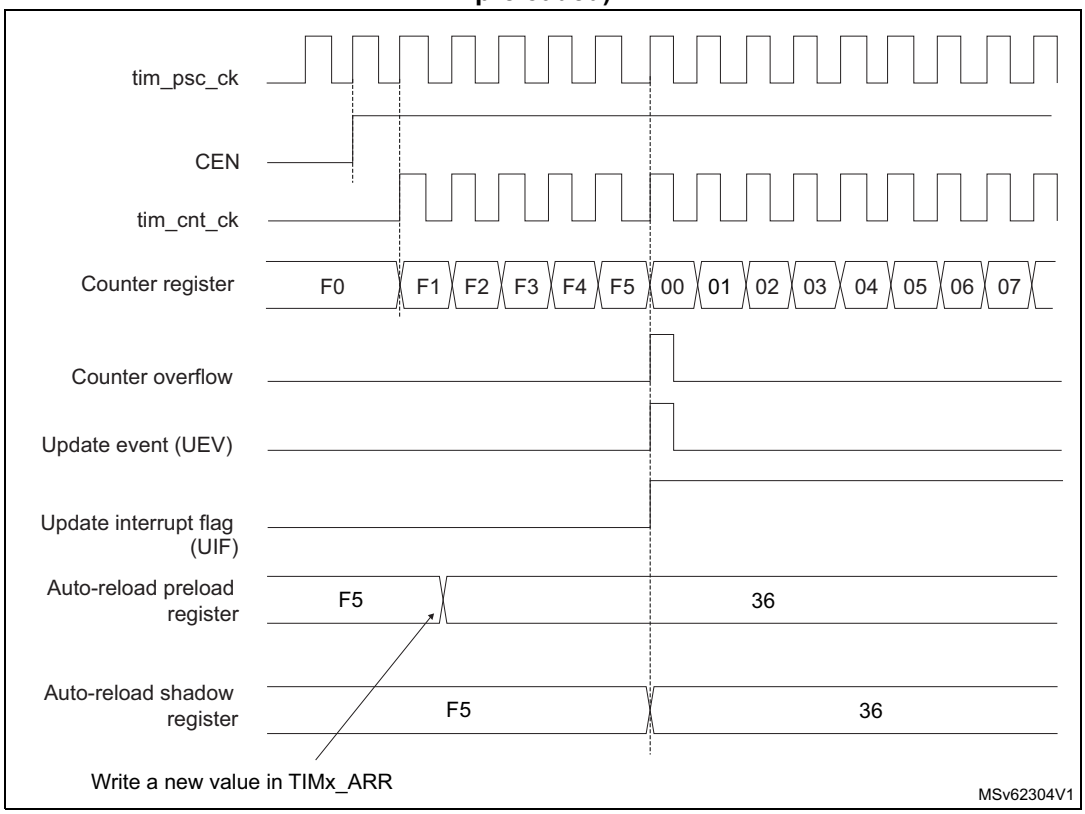


Figure 642. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



56.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

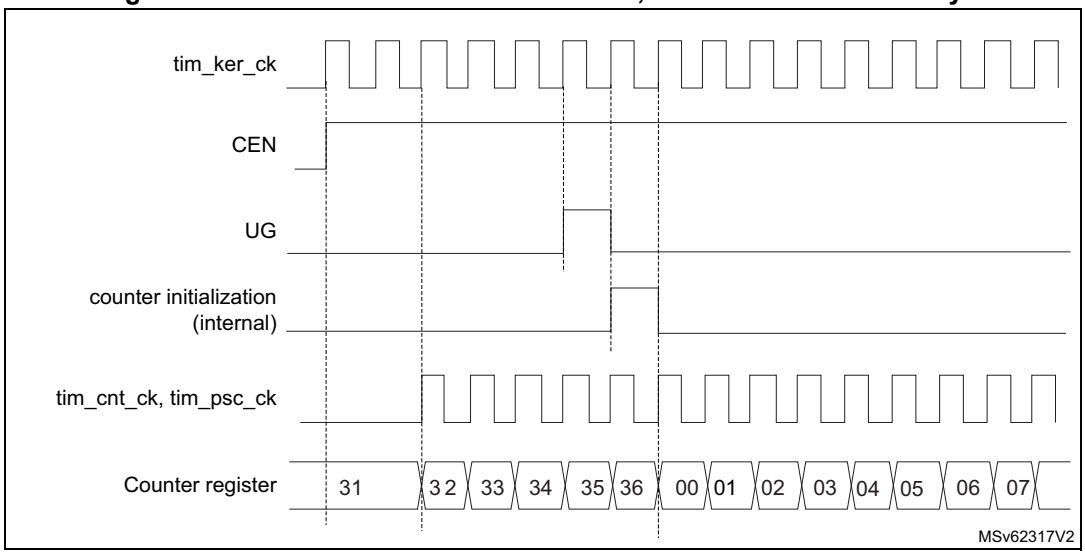
Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM10/TIM11/TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 643 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

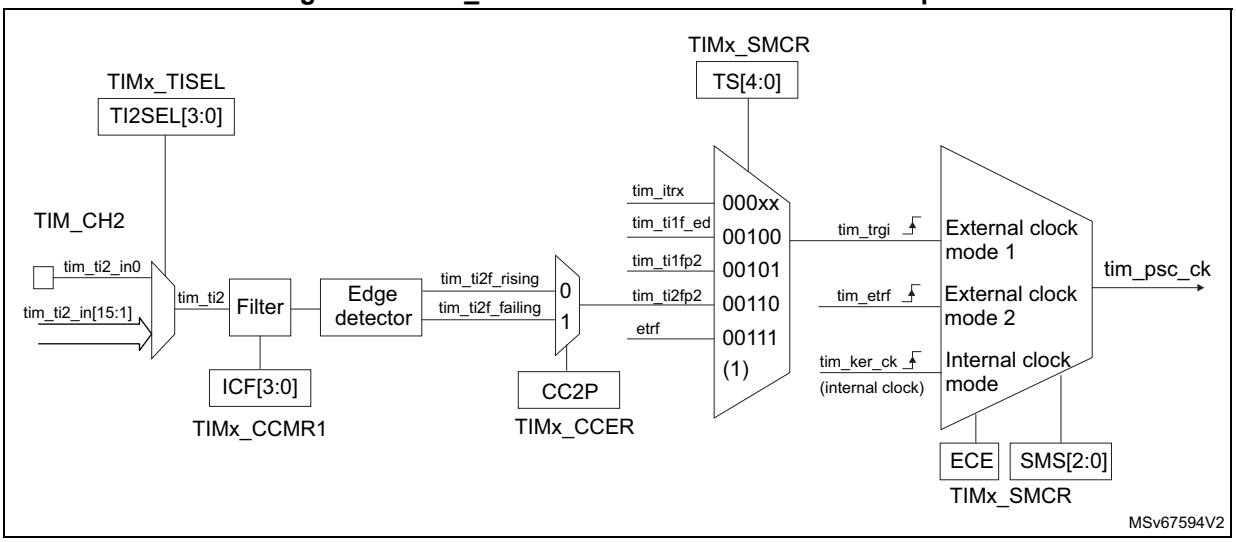
Figure 643. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 644. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

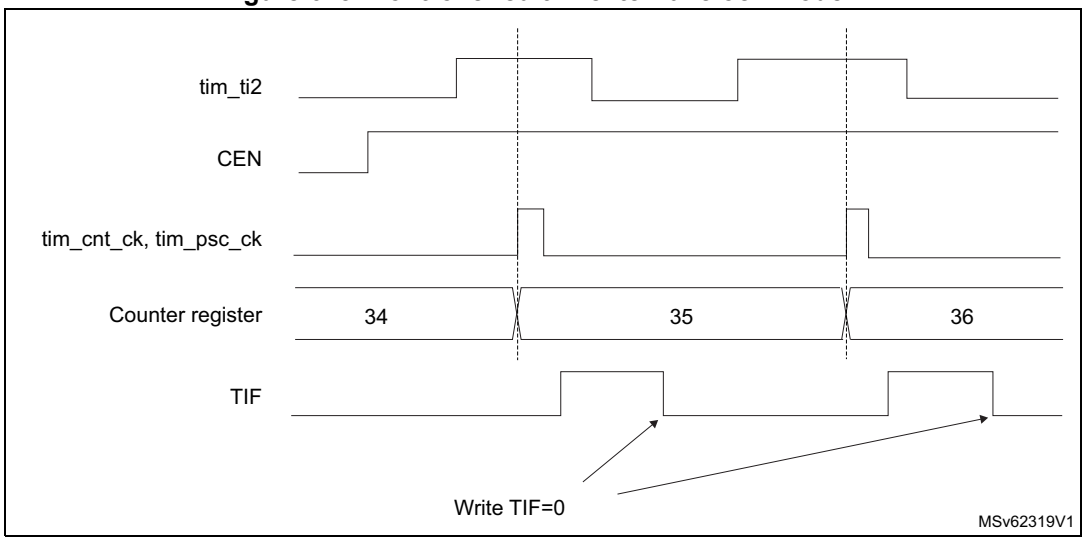
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 645. Control circuit in external clock mode 1



56.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 646, *Figure 647*, *Figure 648* and *Figure 649* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which

can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 646. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14)

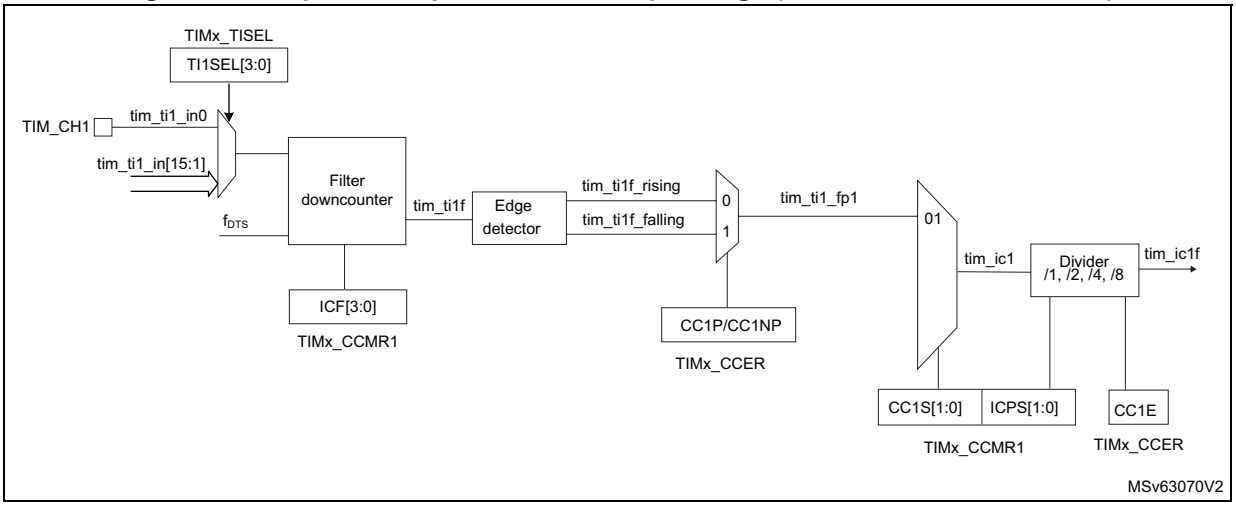
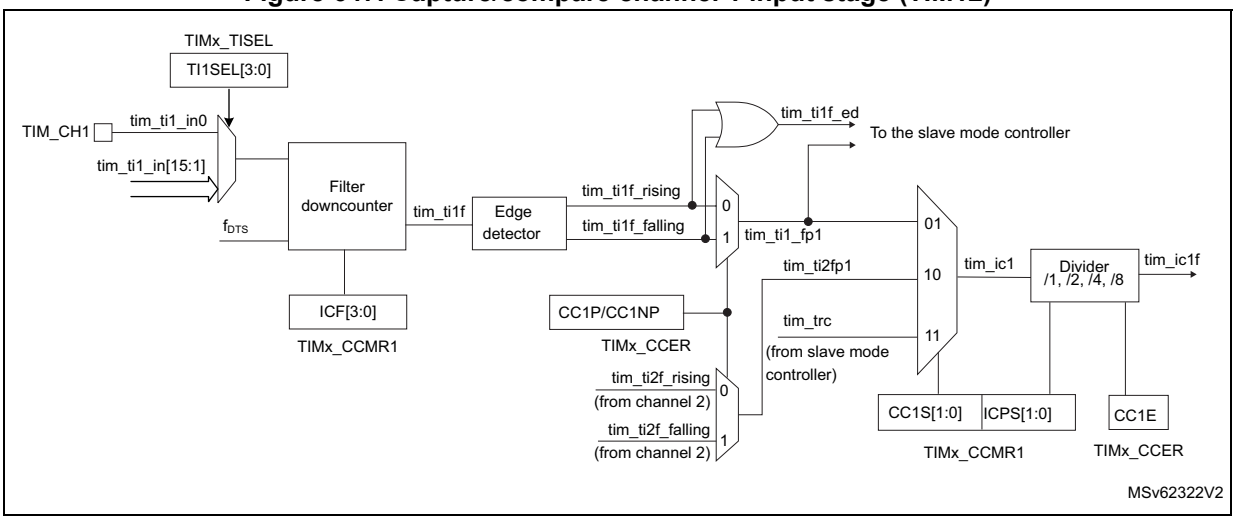


Figure 647. Capture/compare channel 1 input stage (TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 648. Capture/compare channel 1 main circuit

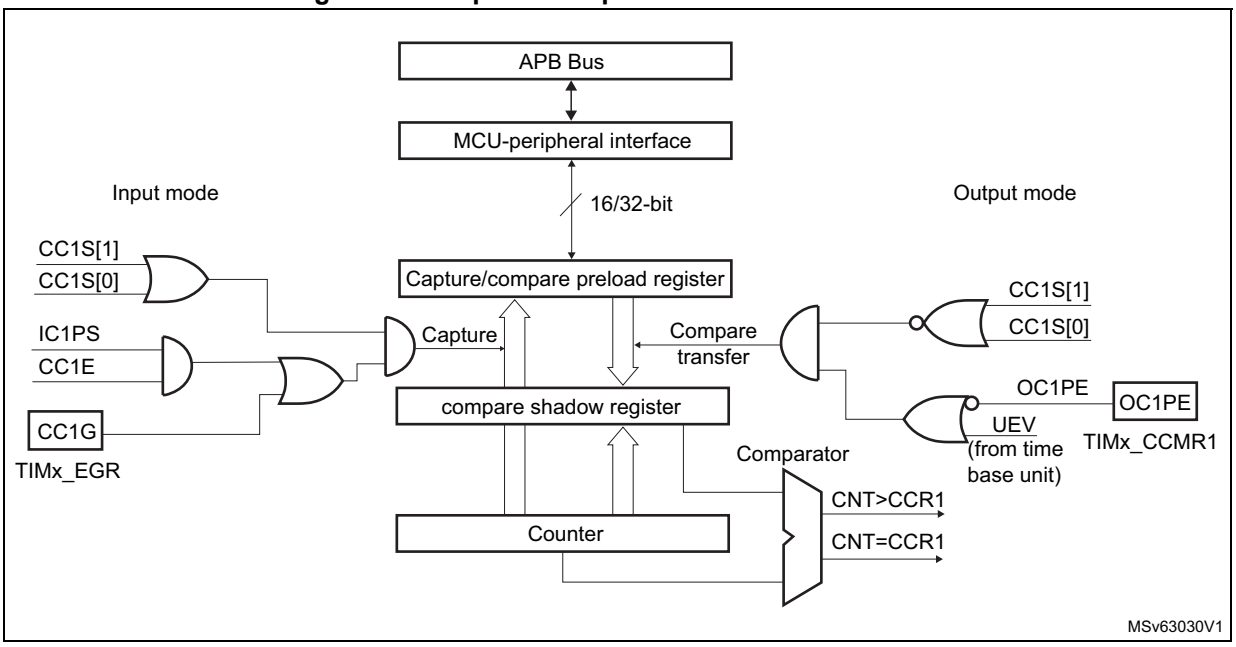
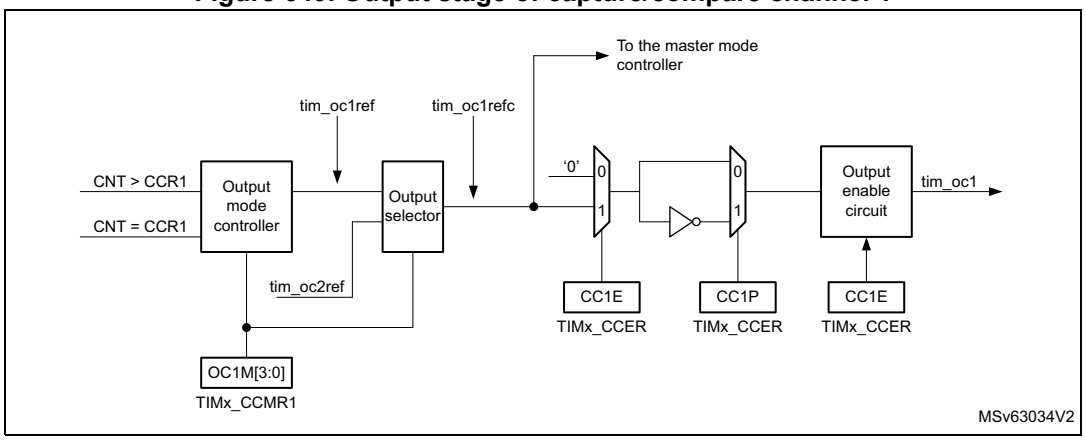


Figure 649. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

56.4.7 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be

cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

56.4.8 PWM input mode (TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

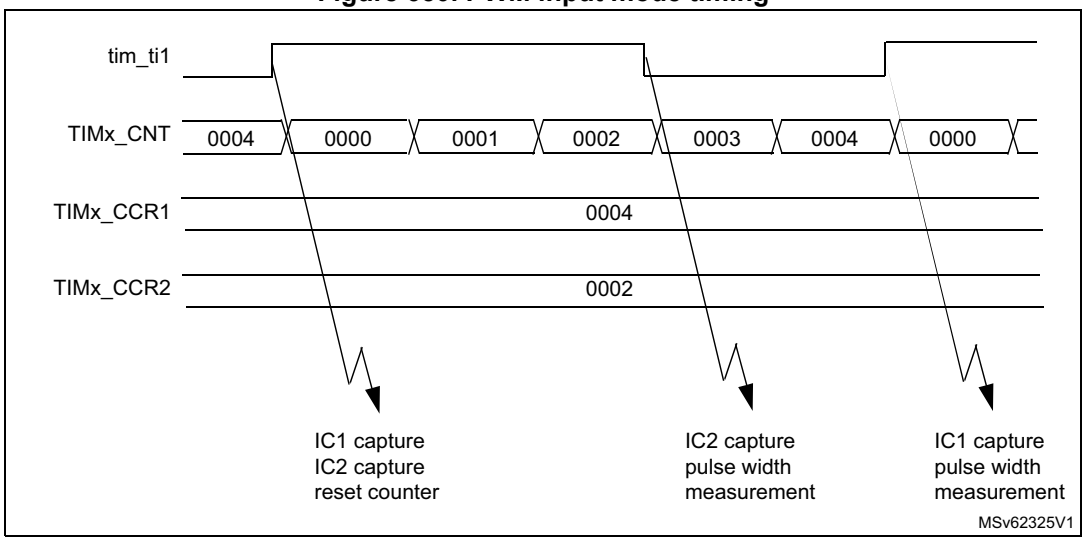
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 650. PWM input mode timing



56.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

56.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

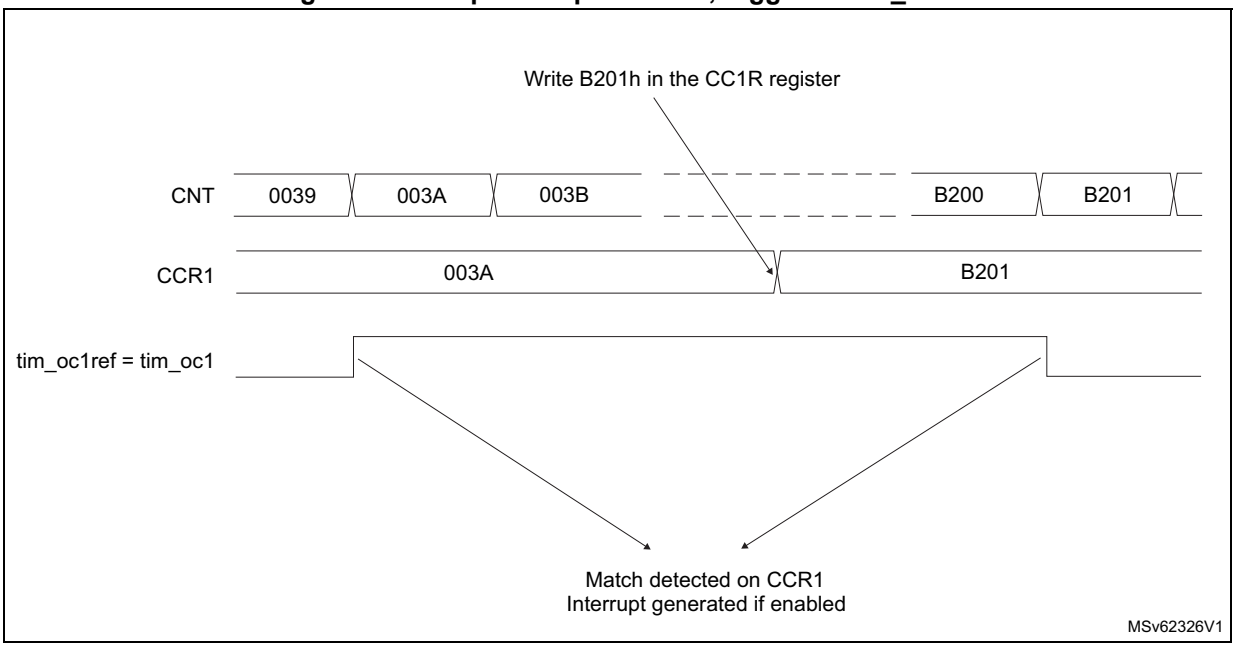
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 651*.

Figure 651. Output compare mode, toggle on tim_oc1.



56.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

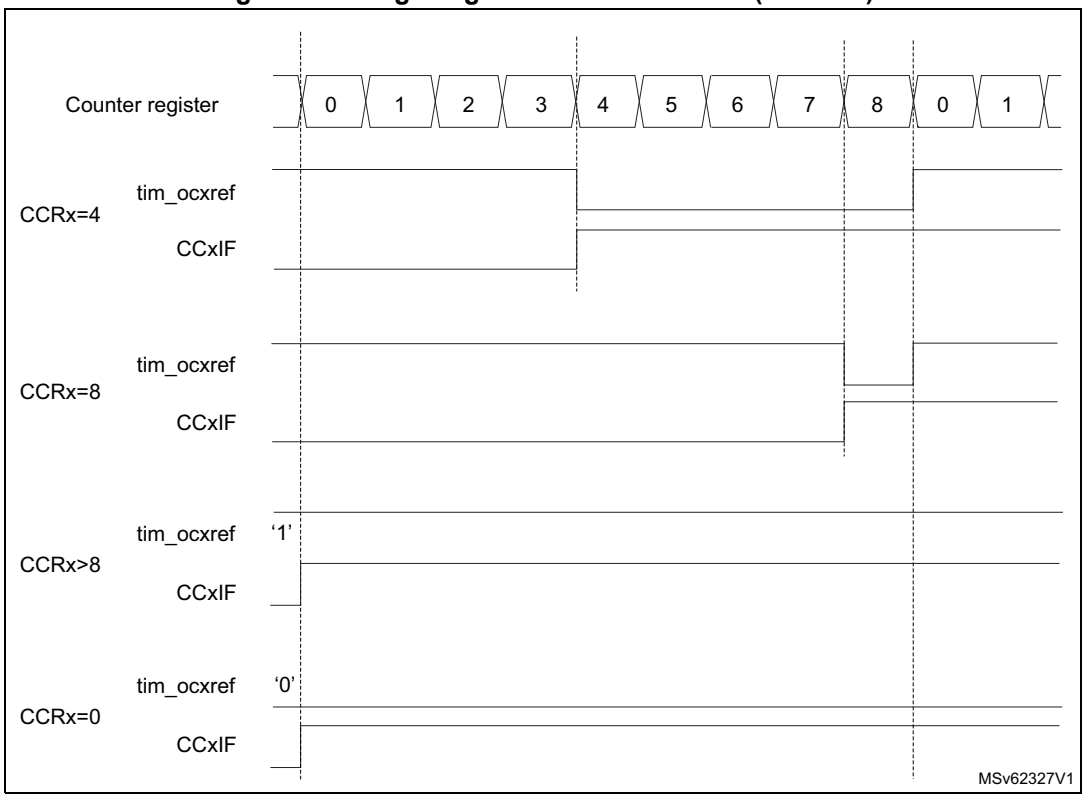
The tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 652* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 652. Edge-aligned PWM waveforms (ARR = 8)

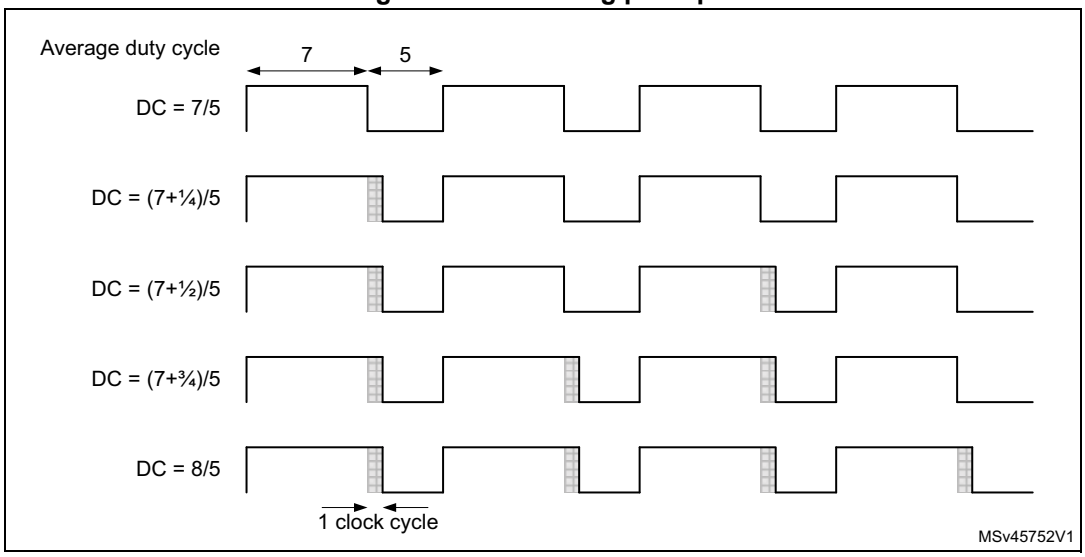


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 653* presents the dithering principle applied to four consecutive PWM cycles.

Figure 653. Dithering principle

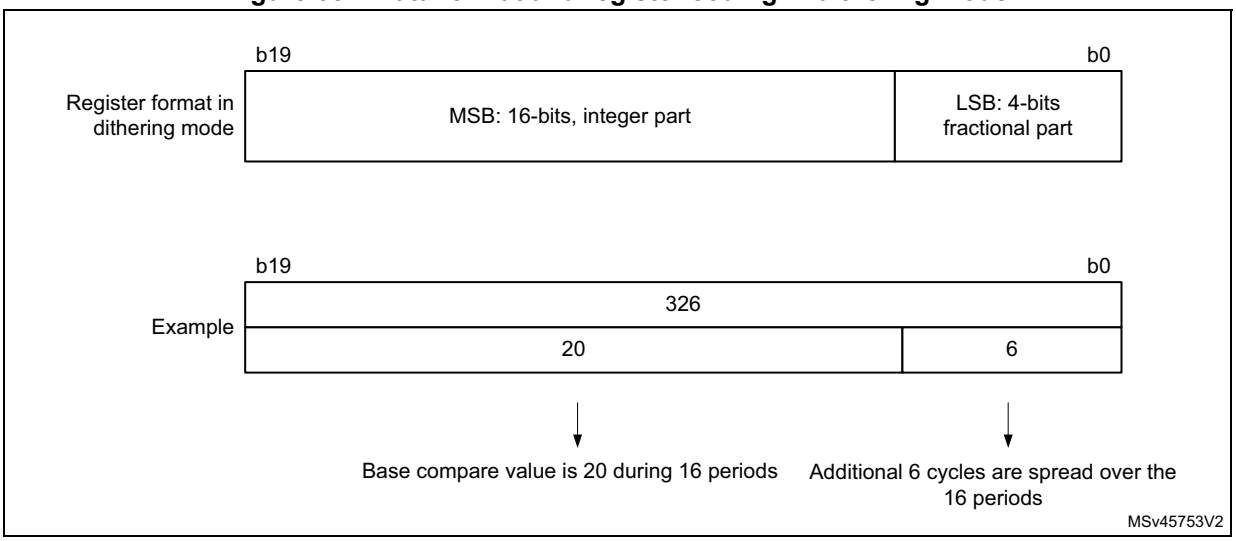


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 654* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 654. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

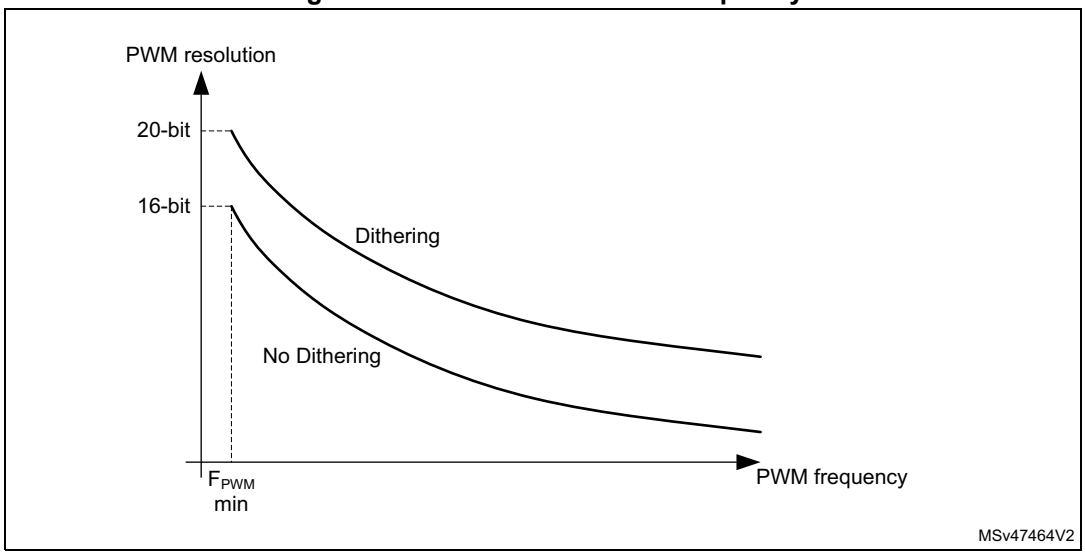
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

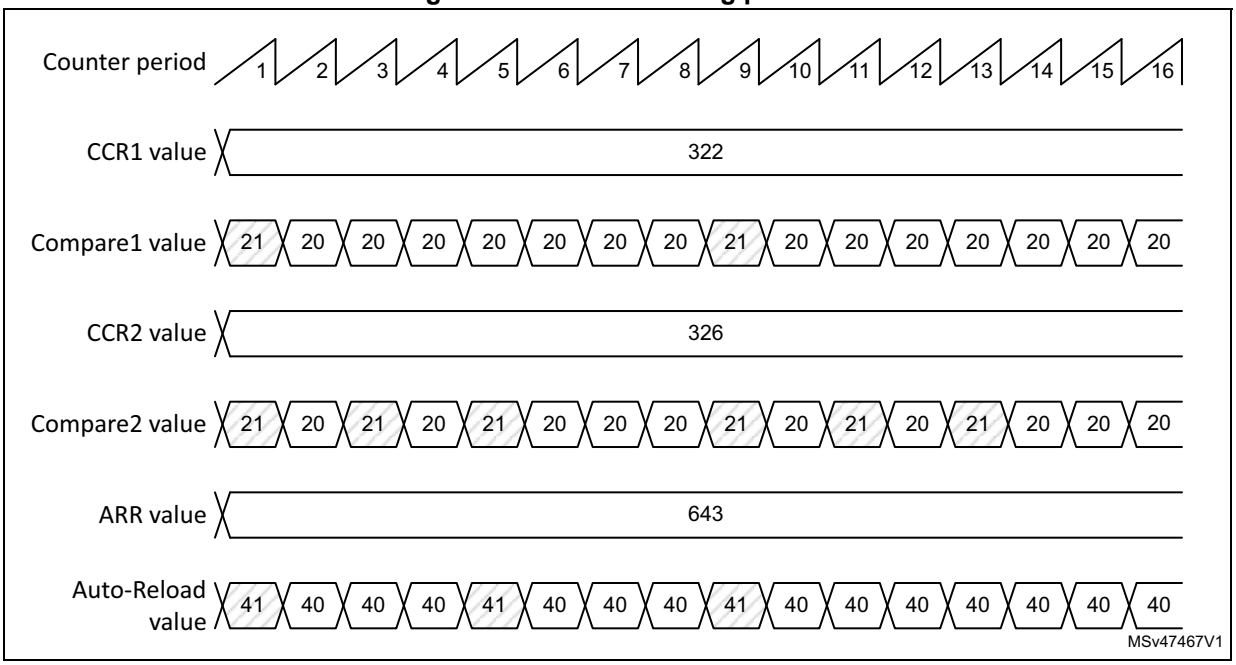
As shown on *Figure 655*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 655. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 656*.

Figure 656. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 569*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 569. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 569. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

56.4.12 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

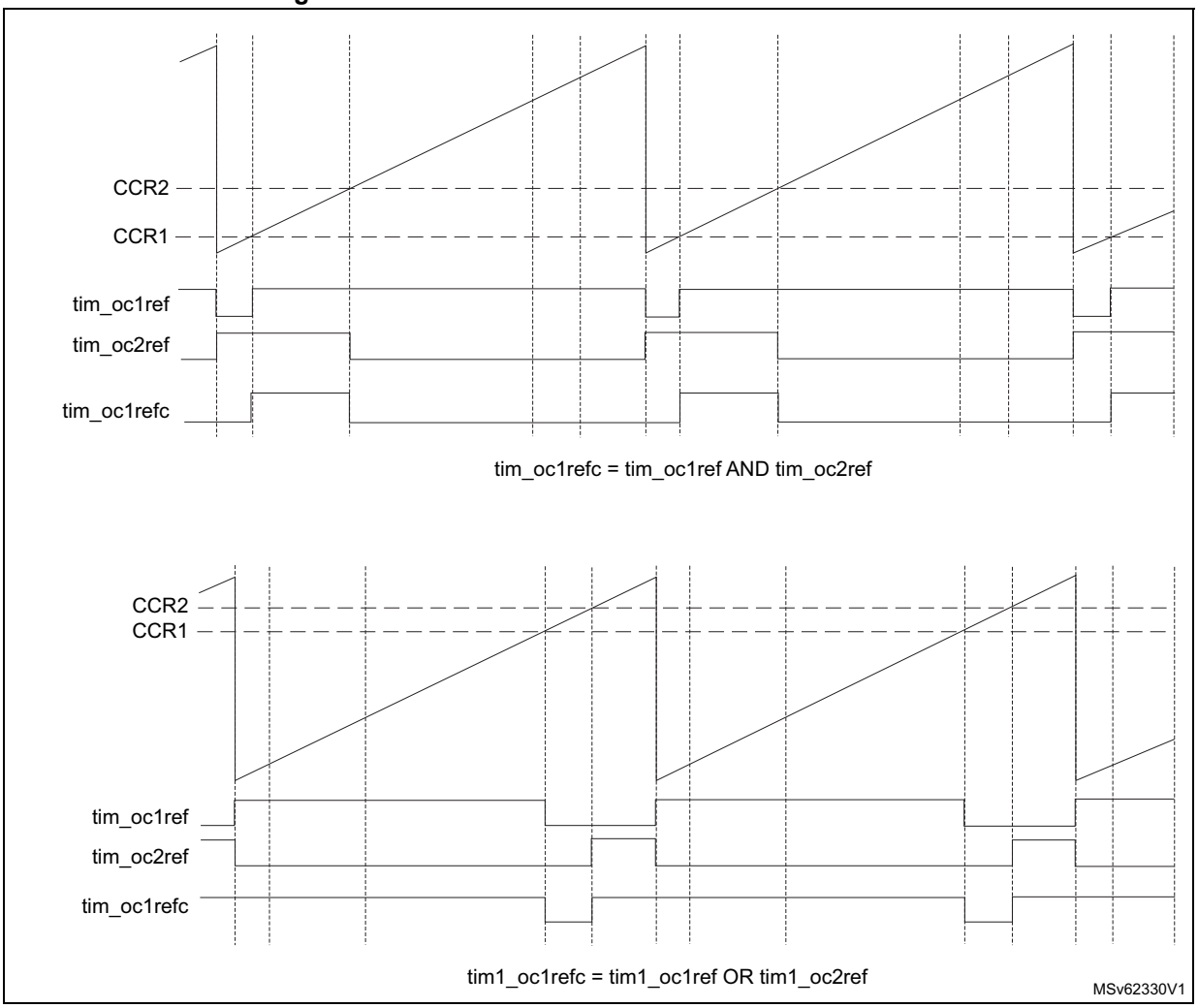
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 657 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 657. Combined PWM mode on channel 1 and 2



56.4.13 One-pulse mode

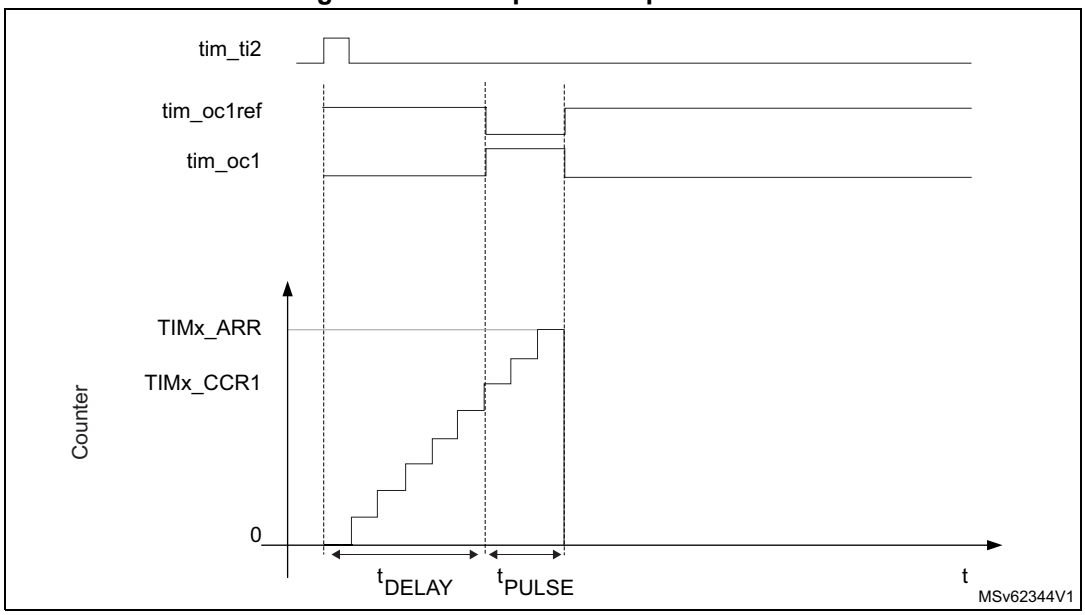
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 658. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

56.4.14 Retriggerable one pulse mode (TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 56.4.13: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

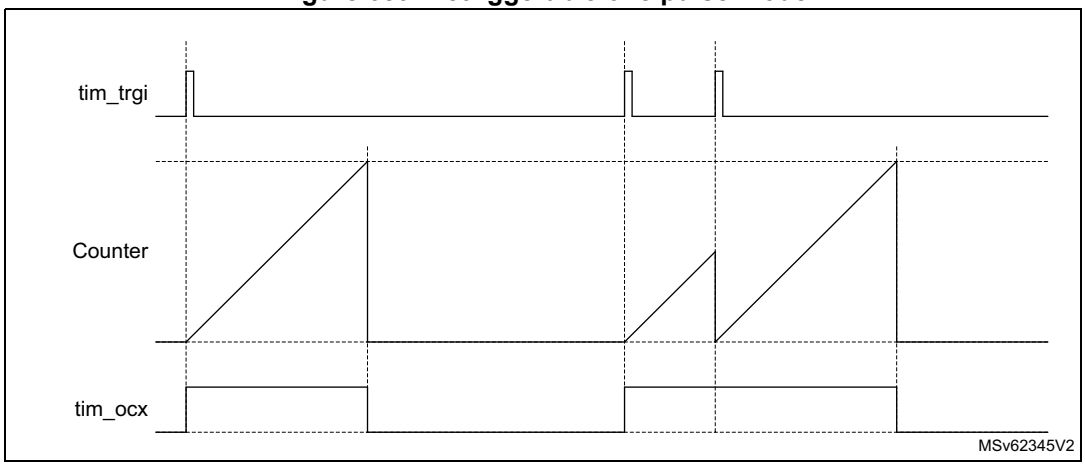
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 659. Retriggerable one pulse mode



56.4.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

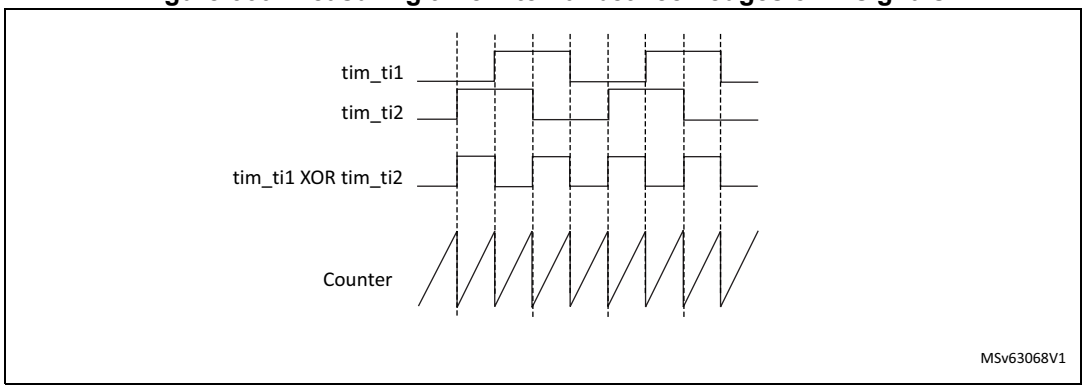
There is no latency between the assertions of the UIF and UIFCPY flags.

56.4.16 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 660*.

Figure 660. Measuring time interval between edges on 2 signals



56.4.17 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the

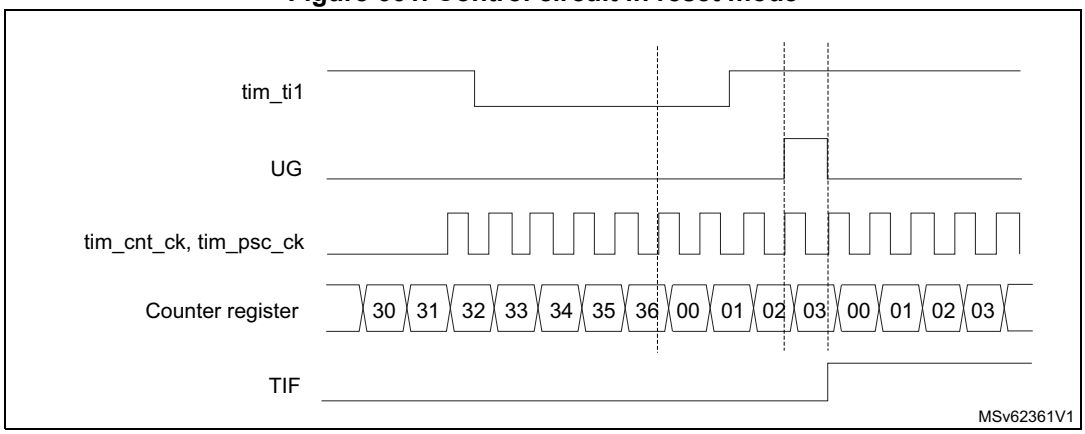
TIMx_CCMR1 register. Program CC1P and CC1NP to 00 in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 661. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

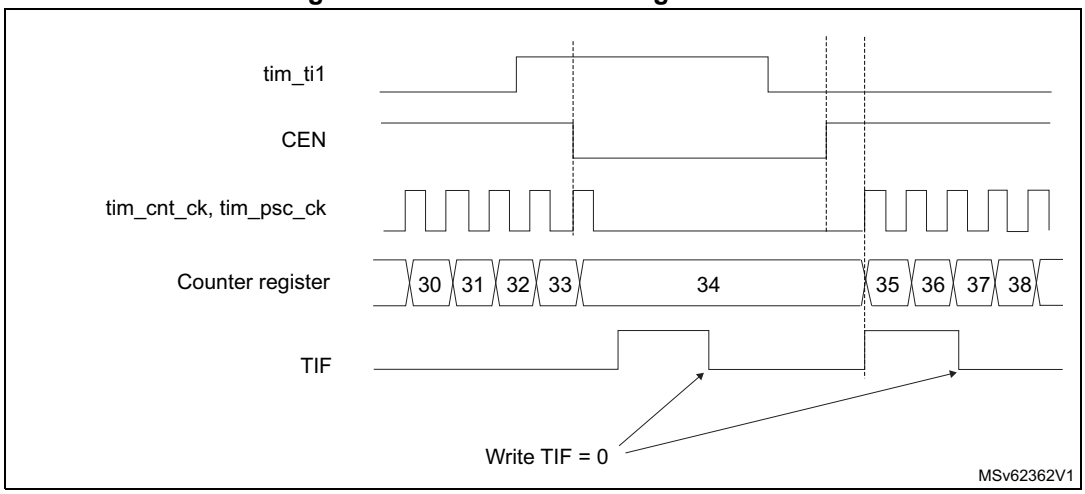
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 662. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

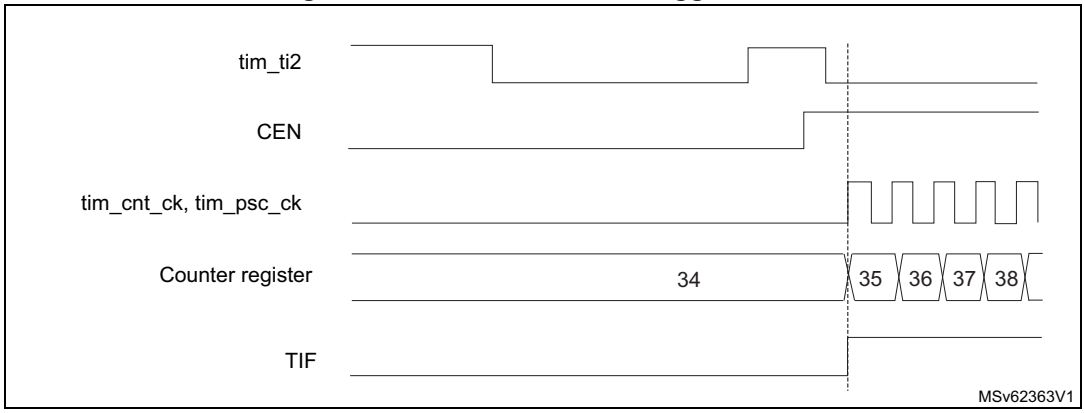
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 663. Control circuit in trigger mode



56.4.18 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

56.4.19 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

56.4.20 Timer synchronization (TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

56.4.21 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

56.4.22 ADC triggers (TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

56.4.23 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

56.5 TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes

Table 570. Effect of low-power modes on TIM10/TIM11/TIM12/TIM13/TIM14

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

56.6 TIM10/TIM11/TIM12/TIM13/TIM14 interrupts

The TIM10/TIM11/TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in *Table 571*.

Table 571. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No

1. Available for TIM12 only.

56.7 TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

56.7.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

56.7.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x004

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

56.7.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal Trigger 10 (tim_itr11)
10000: Internal Trigger 10 (tim_itr12)
10001: Internal Trigger 10 (tim_itr13)
10010: Internal Trigger 10 (tim_itr14)
10011: Internal Trigger 10 (tim_itr15)
Others: Reserved

See *Table 568: TIMx internal trigger connection* for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

56.7.4 TIM12 interrupt enable register (TIM12_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

56.7.5 TIM12 status register (TIM12_SR)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 56.7.3: TIM12 slave mode control register (TIM12_SMCR)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

56.7.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

56.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.7.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00:CC1 channel is configured as output

01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.7.9 TIM12 capture/compare enable register (TIM12_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity

Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity

Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable

Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity

CC1 channel configured as output: CC1NP must be kept cleared

CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 572. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

56.7.10 TIM12 counter (TIM12_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

56.7.11 TIM12 prescaler (TIM12_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

56.7.12 TIM12 autoreload register (TIM12_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 56.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

56.7.13 TIM12 capture/compare register 1 (TIM12_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

56.7.14 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

56.7.15 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIMx_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

0100: tim_ti2_in15

Refer to *Table 567: Interconnect to the tim_ti2 input multiplexer* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIMx_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Table 566: Interconnect to the tim_ti1 input multiplexer* for interconnects list.

56.7.16 TIM12 hardware configuration 2 register (TIM12_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width
0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width
0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear
0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

56.7.17 TIM12 hardware configuration 1 register (TIM12_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x1000 0102

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
0x0: Down-counter only
0x1: Up-counter only
0x2: Up/Down counter allowing center-aligned mode
Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
0x0: DMA requests are not possible
0x1: Single DMA requests possible
0x2: All DMA features implemented
Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
0x0: no break implemented
0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
0x0: External trigger circuit not implemented, ocrefx cleared by ocref_clr input
0x1: External trigger circuit implemented, ocrefx cleared by ocref_clr input
0x2: External trigger circuit implemented, ocrefx cleared by etrf signal
0x3: External trigger circuit implemented, ocrefx cleared by ocref_clr or etrf (1 control bit "OCCS" added in GPT_SMCR)
Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
0x0: repetition counter not implemented
0x1: 8-bit repetition counter implemented
0x2: 16-bit repetition counter implemented
Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

56.7.18 TIM12 IP version register (TIM12_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

56.7.19 TIM12 IP identification register (TIM12_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

56.7.20 TIM12 size ID register (TIM12_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

56.7.21 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 573. TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM12_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004	TIM12_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x008	TIM12_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	TIM12_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x010	TIM12_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x014	TIM12_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	TIM12_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM12_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	TIM12_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	TIM12_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIM12_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 573. TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	TIM12_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	TIM12_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIM12_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	TIM12_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															
0x3EC	TIM12_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_ WIDTH16[3:0]				PSC_ LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIM12_ HWCFGR1	CNT_ DIR[3:0]				DMA_ IMPL[3:0]				BK_ IMPL[3:0]				ETR_ IMPL[3:0]				RCR_ IMPL[3:0]				NB_OF_ SM[3:0]				NB_OF_ DT[3:0]				NB_OF_ CC[3:0]			
	Reset value	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0x3F4	TIM12_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIM12_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIM15_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

56.8 TIM10/TIM11/TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

56.8.1 TIMx control register 1 (TIMx_CR1)(x = 10, 11, 13, 14)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

– Counter overflow

– Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

56.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 10, 11, 13, 14)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

56.8.3 TIMx status register (TIMx_SR)(x = 10, 11, 13, 14)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

56.8.4 TIMx event generation register (TIMx_EGR)(x = 10, 11, 13, 14)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

56.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 10, 11, 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 10, 11, 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 10, 11, 13, 14)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 574. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output Disabled (tim_ocx = 0)
1	tim_ocx = tim_ocxref + Polarity

Note: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

56.8.8 TIMx counter (TIMx_CNT)(x = 10, 11, 13, 14)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

56.8.9 TIMx prescaler (TIMx_PSC)(x = 10, 11, 13, 14)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

56.8.10 TIMx autoreload register (TIMx_ARR)(x = 10, 11, 13, 14)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 56.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

56.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 10, 11, 13, 14)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

56.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 10, 11, 13, 14)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Table 566: Interconnect to the tim_ti1 input multiplexer* for interconnects list.

56.8.13 TIMx hardware configuration 2 register (TIMx_HWCFGR2) (x = 10, 11, 13, 14)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width
0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width
0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear
0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

56.8.14 TIMx hardware configuration 1 register (TIMx_HWCFGR1) (x = 10, 11, 13, 14)

Address offset: 0x3F0

Reset value: 0x1000 0001

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
- 0x0: Down-counter only
- 0x1: Up-counter only
- 0x2: Up/Down counter allowing center-aligned mode
- Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
- 0x0: DMA requests are not possible
- 0x1: Single DMA requests possible
- 0x2: All DMA features implemented
- Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
- 0x0: no break implemented
- 0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
- 0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
- 0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
- Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
- 0x0: External trigger circuit not implemented, ocrefx cleared by ocref_clr input
- 0x1: External trigger circuit implemented, ocrefx cleared by ocref_clr input
- 0x2: External trigger circuit implemented, ocrefx cleared by etrf signal
- 0x3: External trigger circuit implemented, ocrefx cleared by ocref_clr or etrf (1 control bit "OCCS" added in GPT_SMCR)
- Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
- 0x0: repetition counter not implemented
- 0x1: 8-bit repetition counter implemented
- 0x2: 16-bit repetition counter implemented
- Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

56.8.15 TIMx IP version register (TIMx_VERR)(x = 10, 11, 13, 14)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

56.8.16 TIMx IP identification register (TIMx_IPIDR)(x = 10, 11, 13, 14)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

56.8.17 TIMx size ID register (TIMx_SIDR)(x = 10, 11, 13, 14)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

56.8.18 TIM10/TIM11/TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 575. TIM10/TIM11/TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004 to 0x008	Reserved	Res.																															
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0								0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x018	TIMx_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															

Table 575. TIM10/TIM11/TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x058	Reserved	Res.																															
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															
0x3EC	TIMx_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_ WIDTH16[3:0]				PSC_ LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIMx_ HWCFGR1	CNT_ DIR[3:0]				DMA_ IMPL[3:0]				BK_ IMPL[3:0]				ETR_ IMPL[3:0]				RCR_ IMPL[3:0]				NB_OF_ SM[3:0]				NB_OF_ DT[3:0]				NB_OF_ CC[3:0]			
	Reset value	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3F4	TIMx_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]			MINREV[3:0]
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIMx_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIMx_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

57 General purpose timers (TIM15/TIM16/TIM17)

57.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 57.4.26: Timer synchronization (TIM15 only)*.

57.2 TIM15 main features

TIM15 includes the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

57.3 TIM16/TIM17 main features

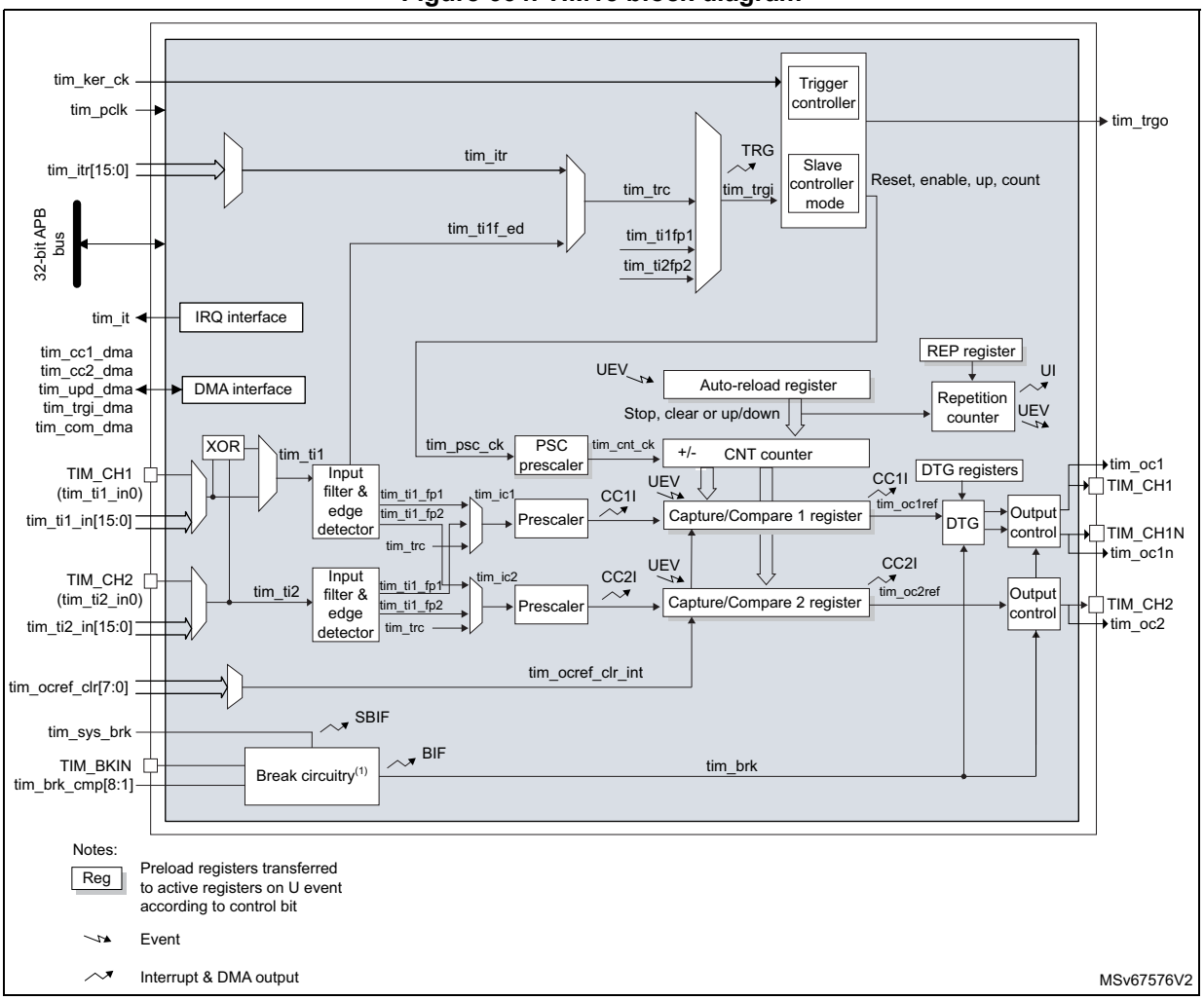
The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

57.4 TIM15/TIM16/TIM17 functional description

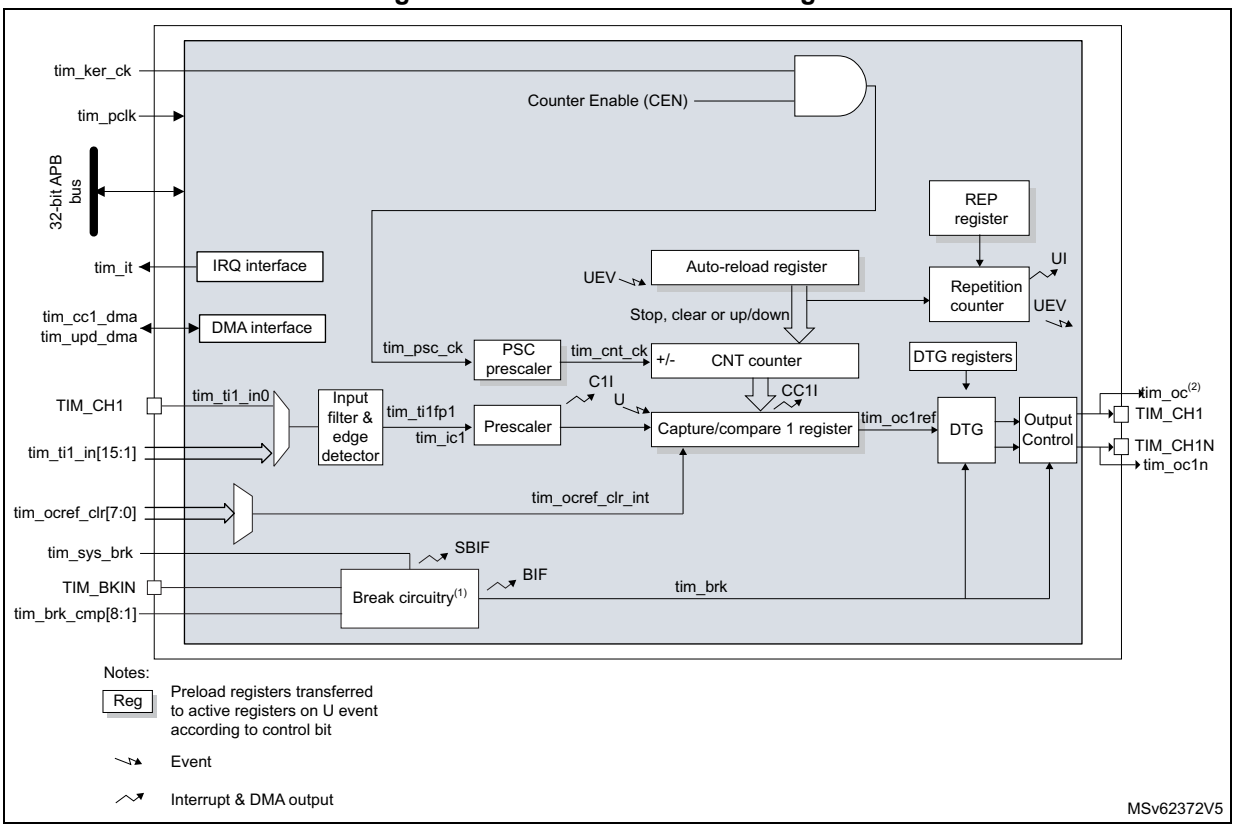
57.4.1 Block diagram

Figure 664. TIM15 block diagram



1. Refer to *Section 57.4.15: Using the break function* for details.

Figure 665. TIM16/TIM17 block diagram



1. Refer to *Section 57.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 57.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

57.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 576 and *Table 577* in this section summarize the TIM inputs and outputs.

Table 576. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 577. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_cc2_dma	Output	Timer capture / compare 2 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Tables below list the sources connected to the tim_ti[2:1] input multiplexers.

Table 578. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	TIM2_CH1	LSI	spdifrx_fs
tim_ti1_in2	TIM3_CH1	LSE	HSE_RTC
tim_ti1_in3	TIM4_CH1	RTC Wake up	MCO1

Table 578. Interconnect to the tim_ti1 input multiplexer (continued)

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in4	LSE	Reserved	Reserved
tim_ti1_in5	CSI		
tim_ti1_in6	MCO2		
tim_ti1_in7	CSI/256		
tim_ti1_in8	HSI/1024		
tim_ti1_in[15:9]	Reserved		

Table 579. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	TIM15 sources
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	TIM2_CH2
tim_ti2_in2	TIM3_CH2
tim_ti2_in3	TIM4_CH2
tim_ti2_in[15:4]	Reserved

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 580. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	tim12_trgo
tim_itr7	tim13_oc1
tim_itr8	tim14_oc1
tim_itr9	Reserved
tim_itr10	tim16_oc1
tim_itr11	tim17_oc1
tim_itr12	Reserved
tim_itr13	tim10_oc1

Table 580. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM15
tim_itr14	tim11_oc1
tim_itr[15]	Reserved

Tables below list the sources connected to the tim_brk input.

Table 581. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	Reserved		
tim_brk_cmp2			
tim_brk_cmp3			
tim_brk_cmp4			
tim_brk_cmp5			
tim_brk_cmp6			
tim_brk_cmp7			
tim_brk_cmp8	mdf1_brk0	mdf1_brk1	mdf1_brk2

Table 582. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SYSCFG_CBR register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	Programmable voltage detector (PVD)	PVDL
tim_sys_brk2	Backup RAM double ECC error	BKRAML
tim_sys_brk3	Retention SRAM double ECC error	RETRAML
tim_sys_brk4	Clock security system (CSS)	None (always enabled)

57.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Table 676 and *Table 667* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 666. Counter timing diagram with prescaler division change from 1 to 2

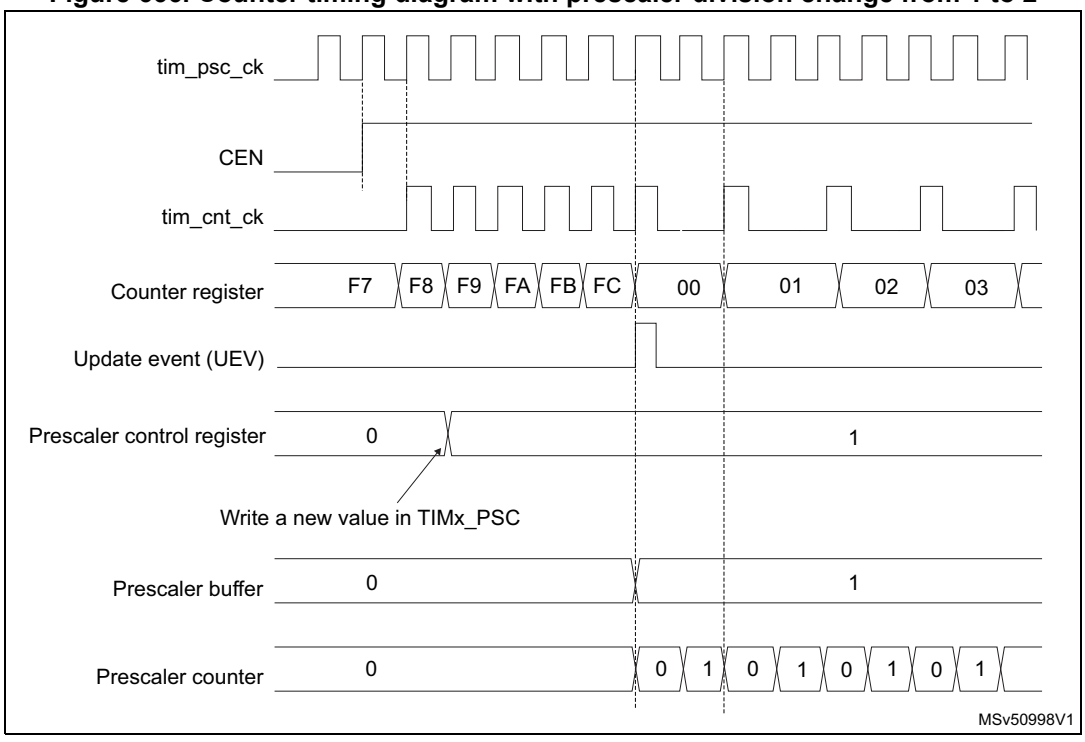
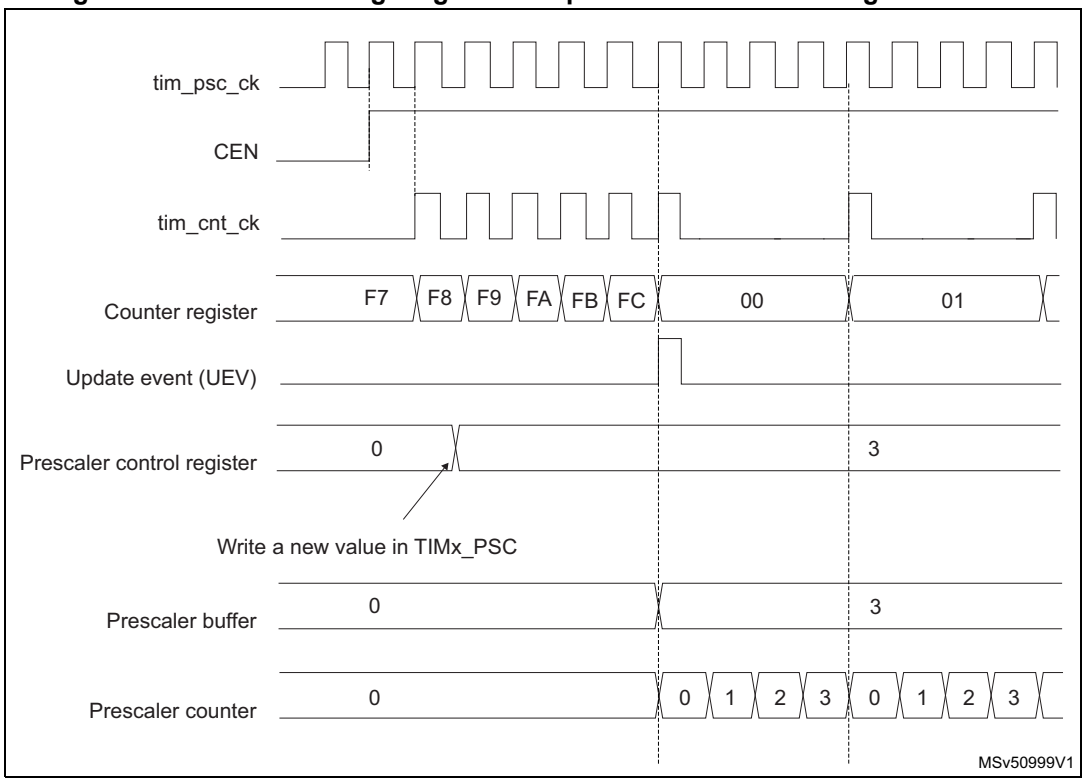


Figure 667. Counter timing diagram with prescaler division change from 1 to 4



57.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 668. Counter timing diagram, internal clock divided by 1

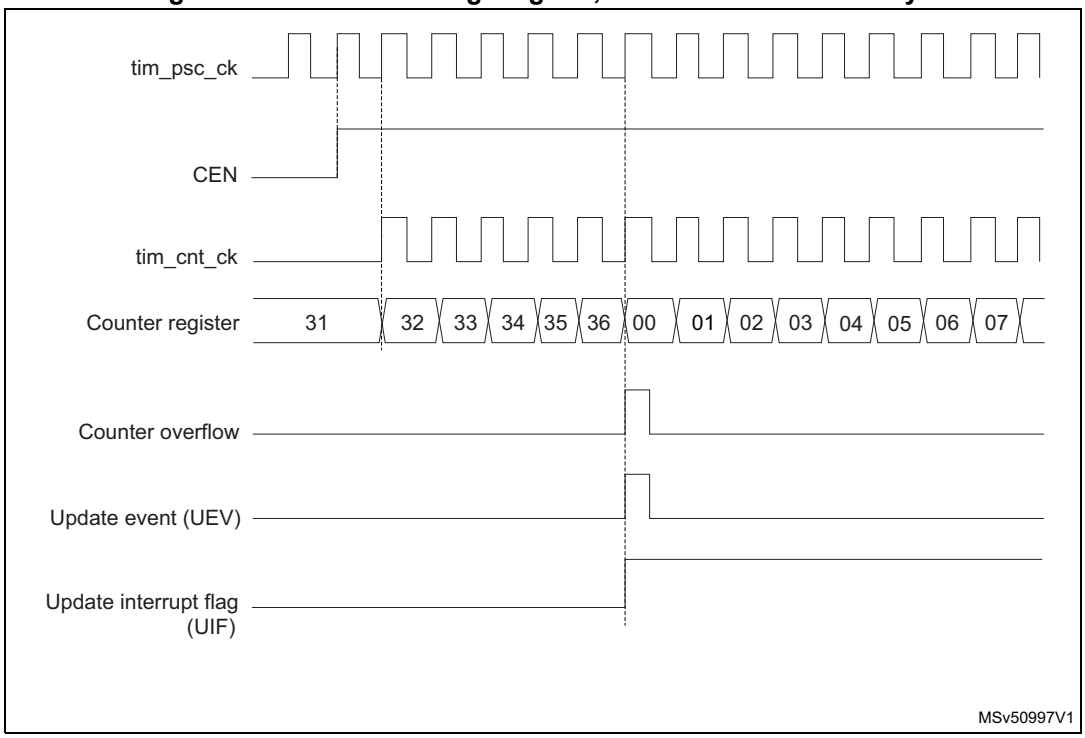


Figure 669. Counter timing diagram, internal clock divided by 2

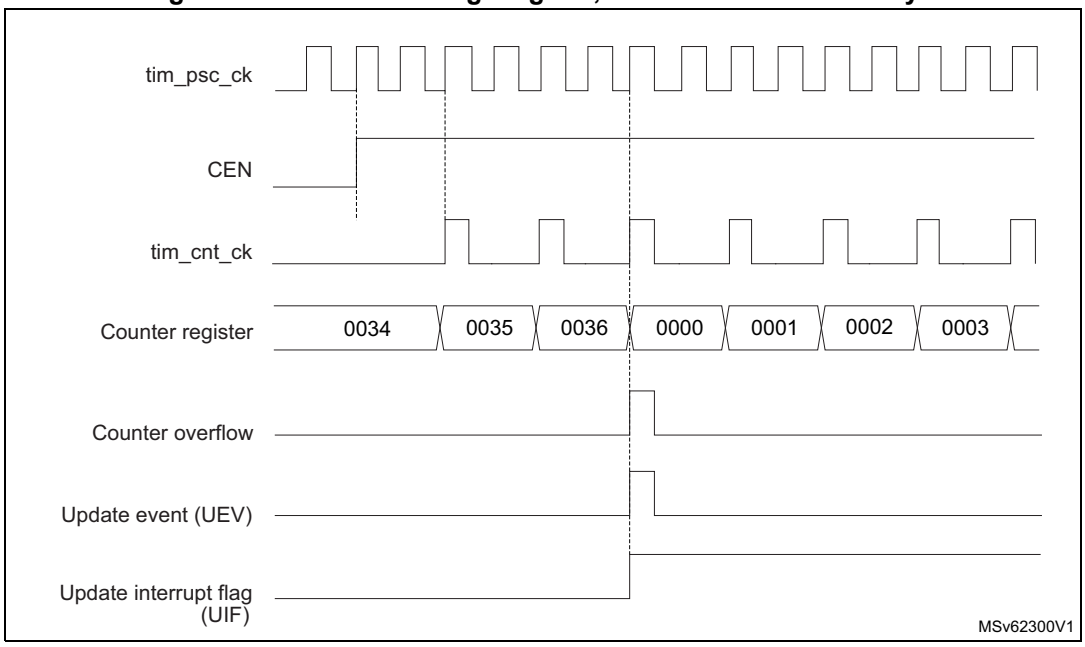


Figure 670. Counter timing diagram, internal clock divided by 4

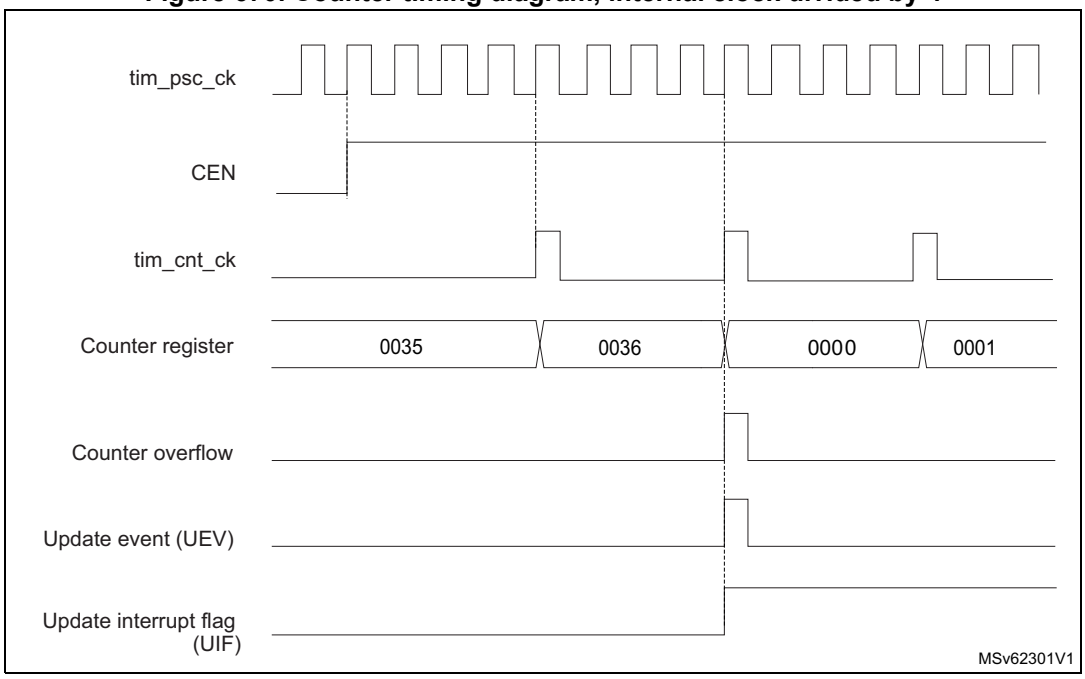


Figure 671. Counter timing diagram, internal clock divided by N

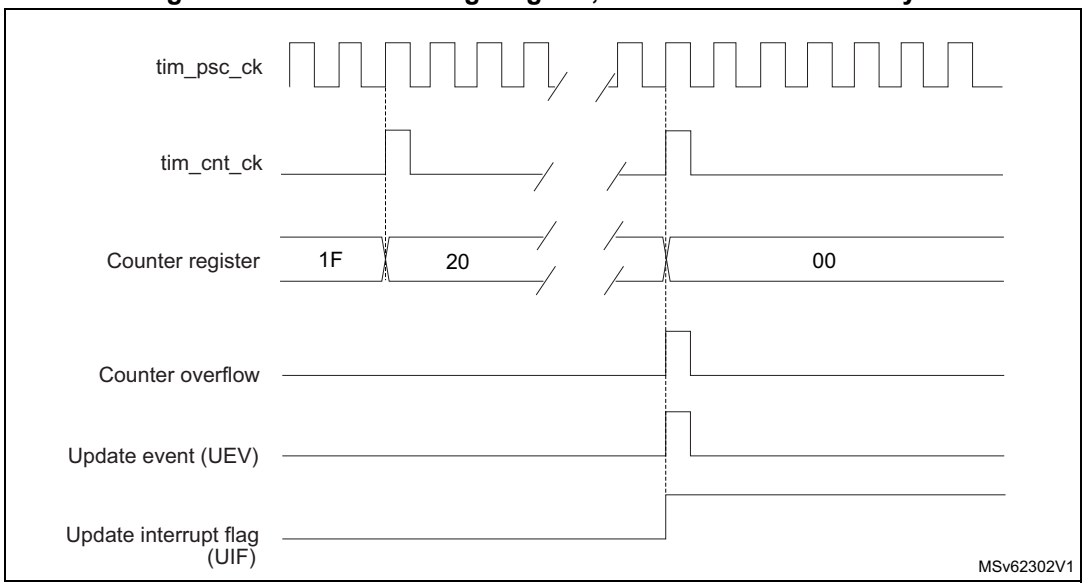


Figure 672. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

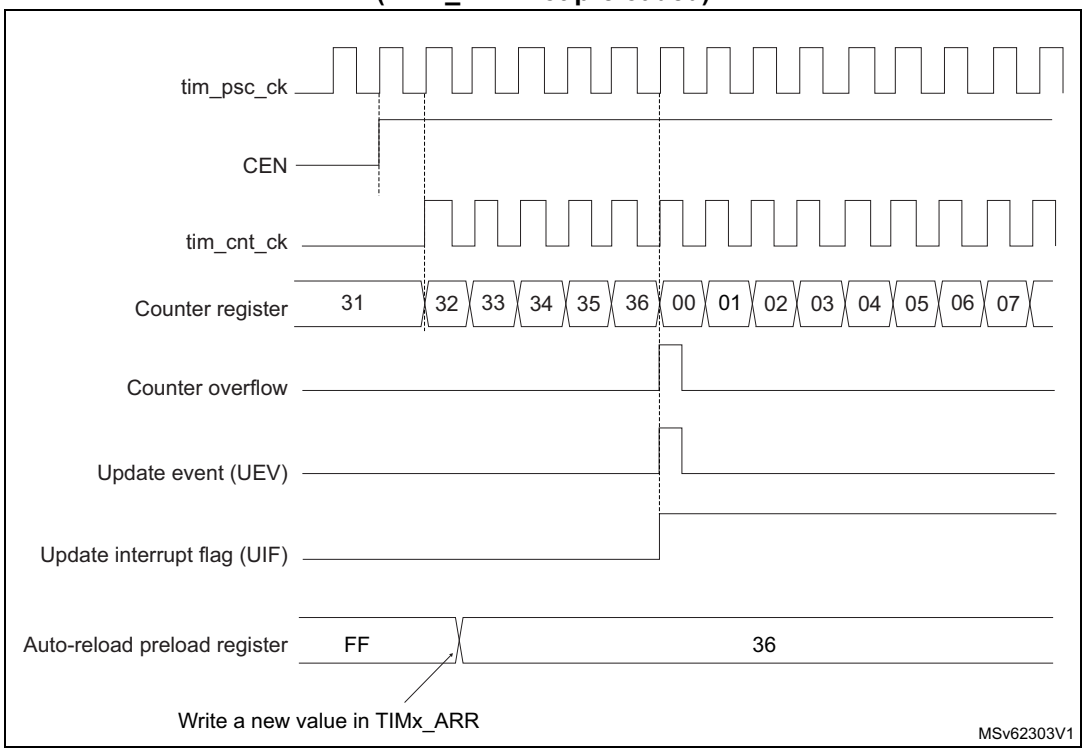
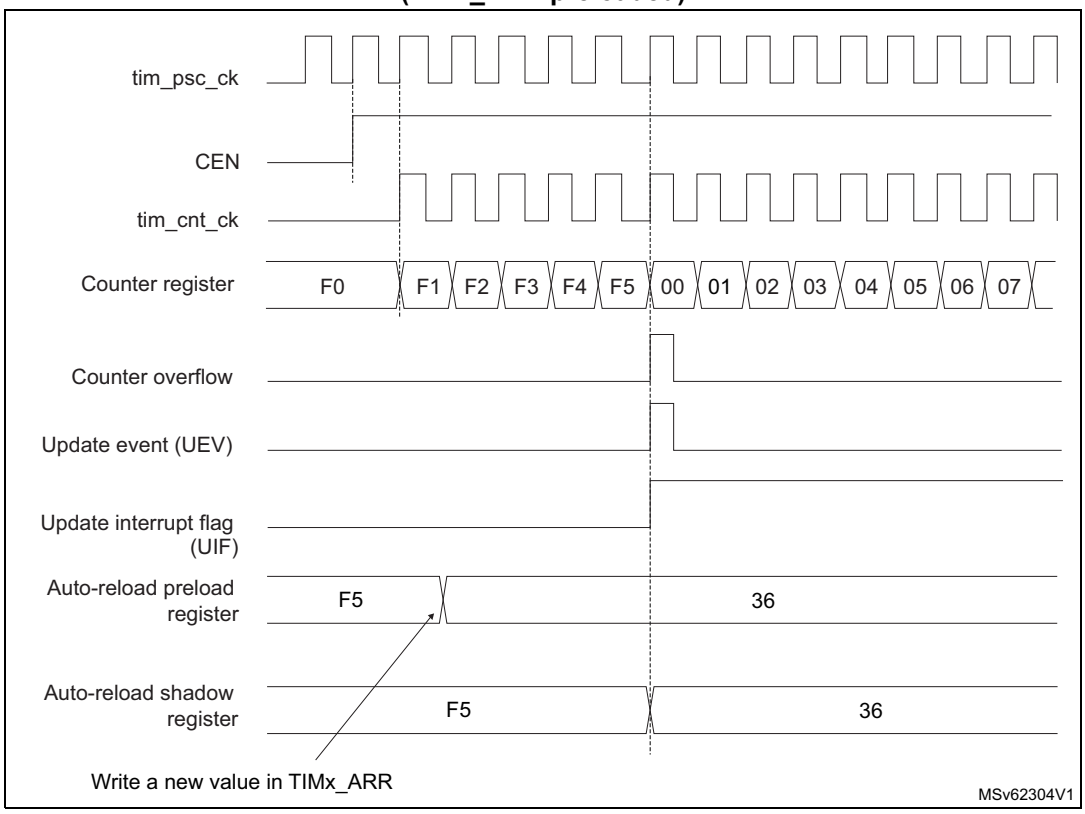


Figure 673. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



57.4.5 Repetition counter

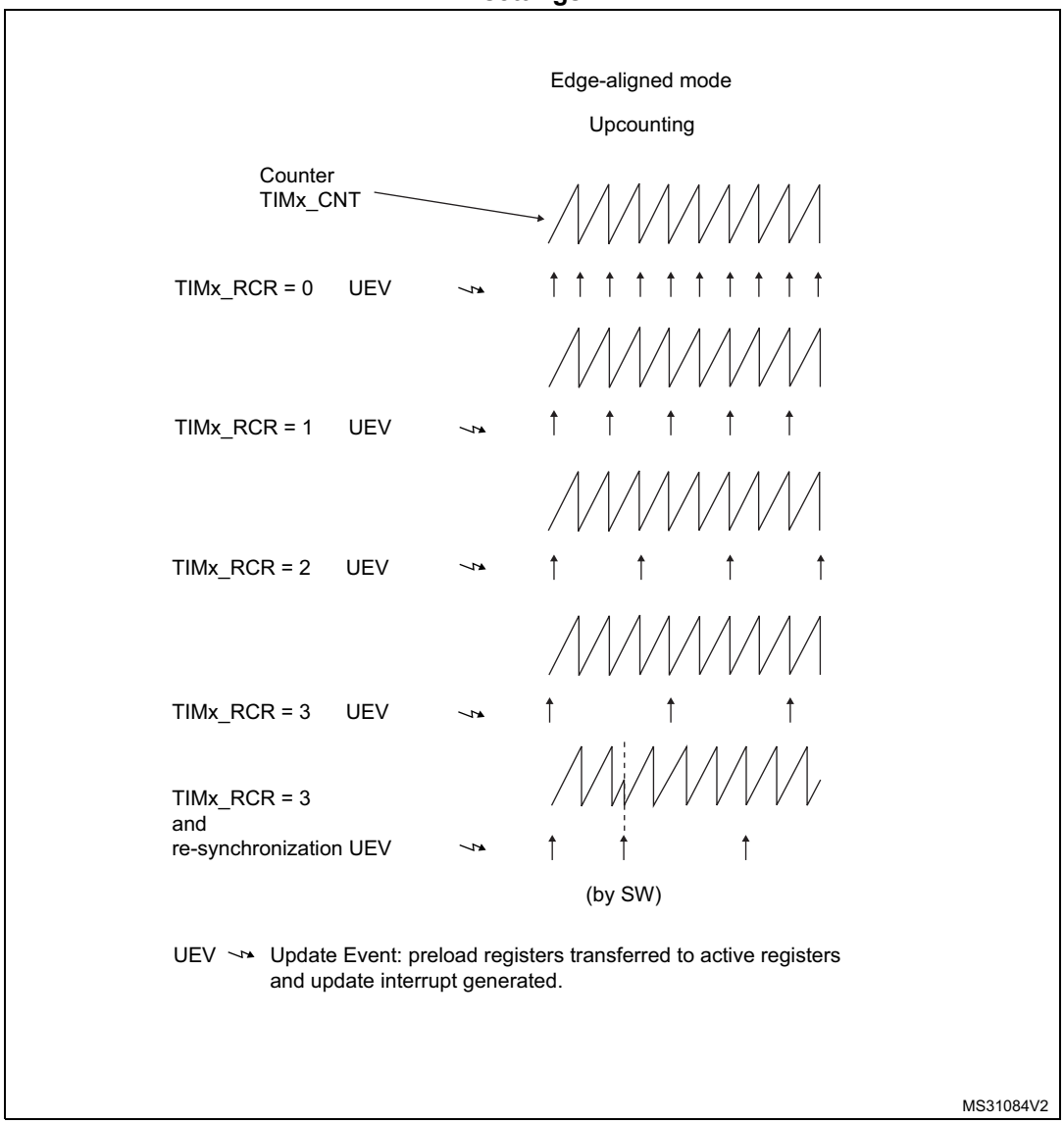
Section 57.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 674*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 674. Update rate examples depending on mode and TIMx_RCR register settings



57.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer* for more details.

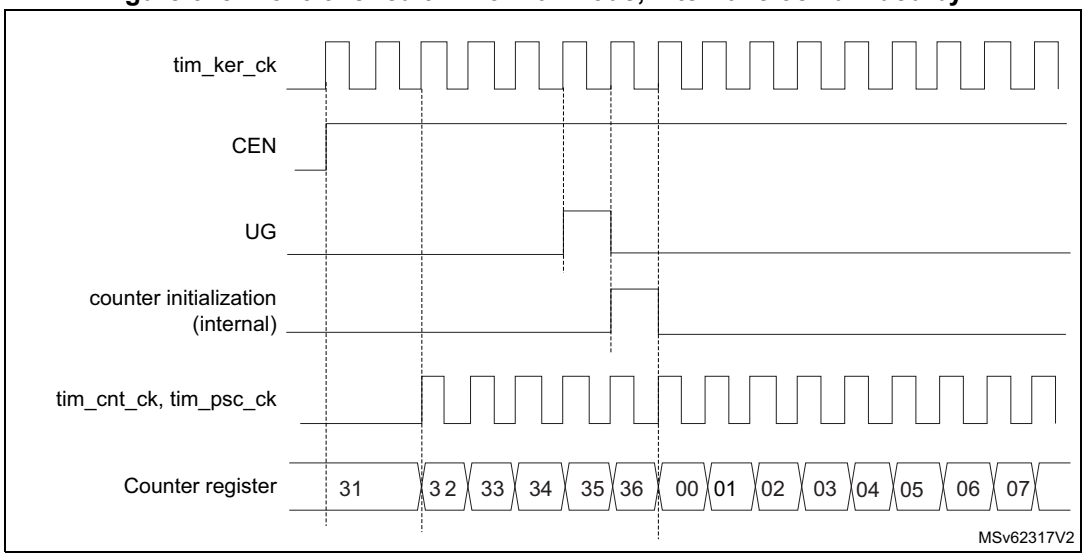
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 675 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

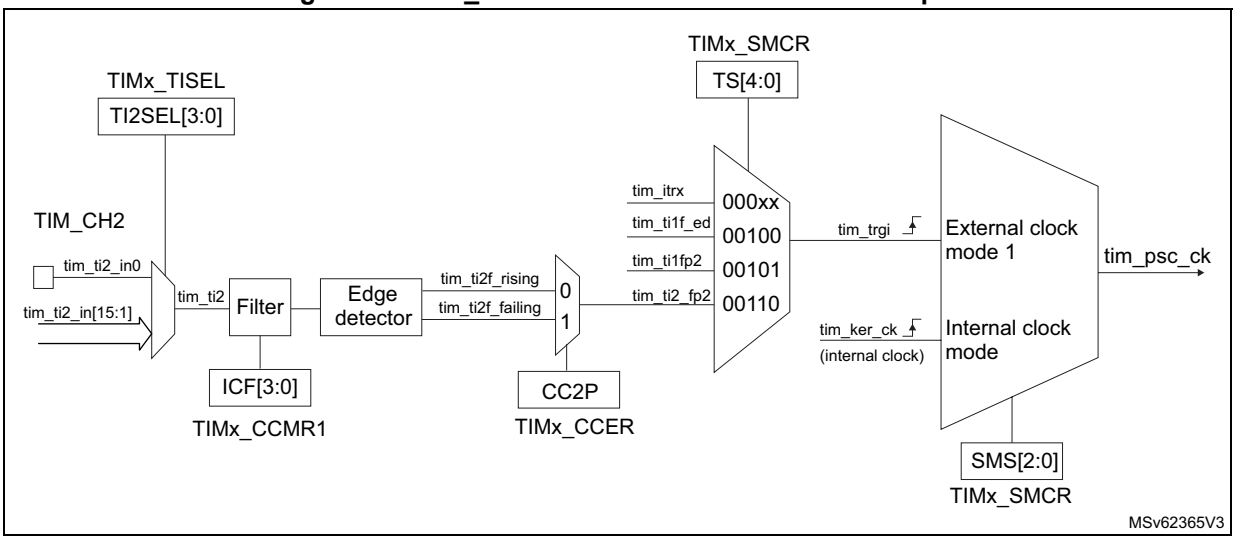
Figure 675. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 676. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

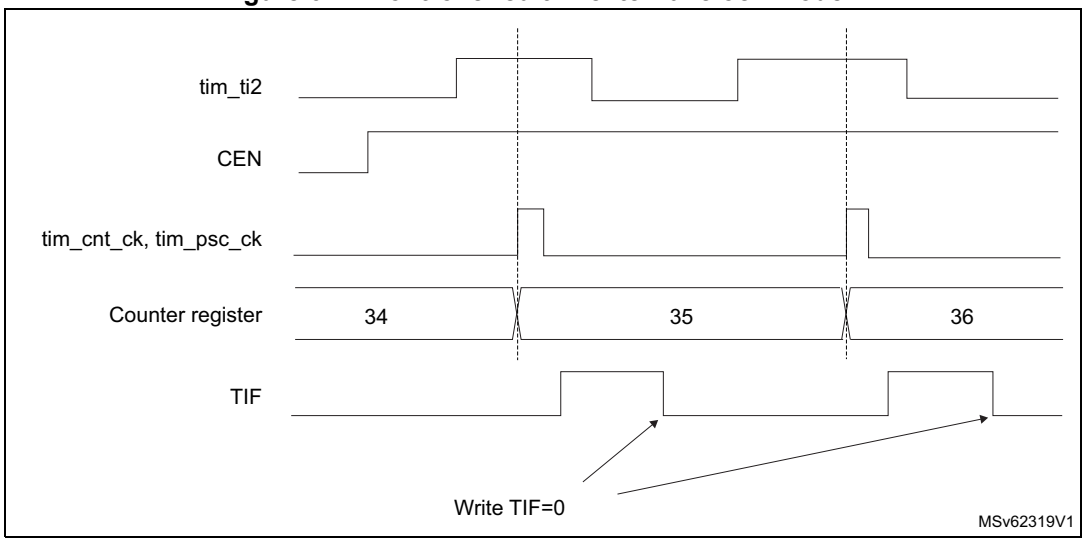
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 677. Control circuit in external clock mode 1



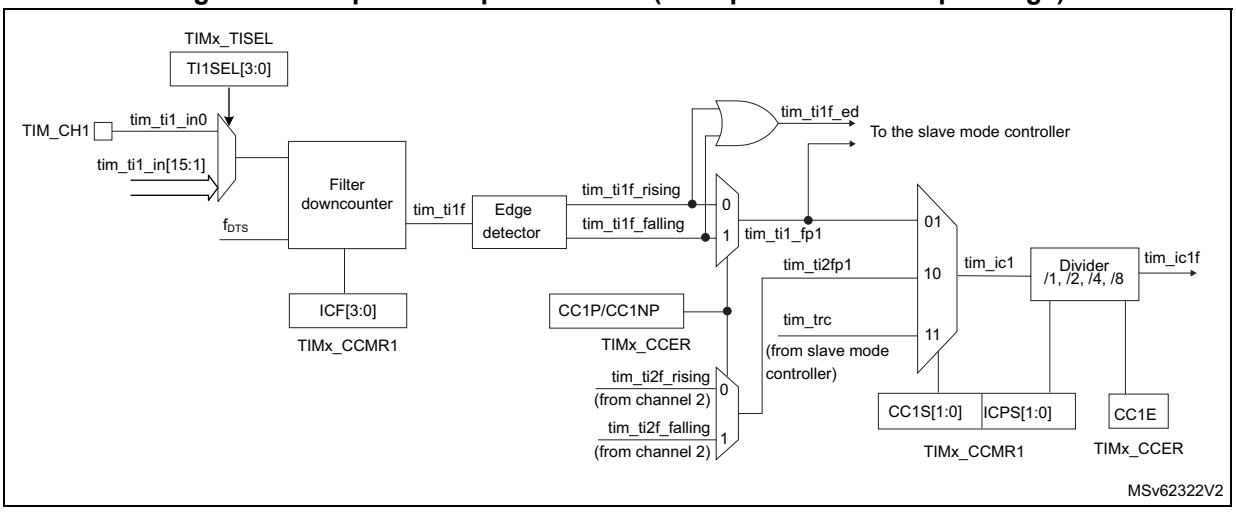
57.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 678 to *Figure 681* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 678. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 679. Capture/compare channel 1 main circuit

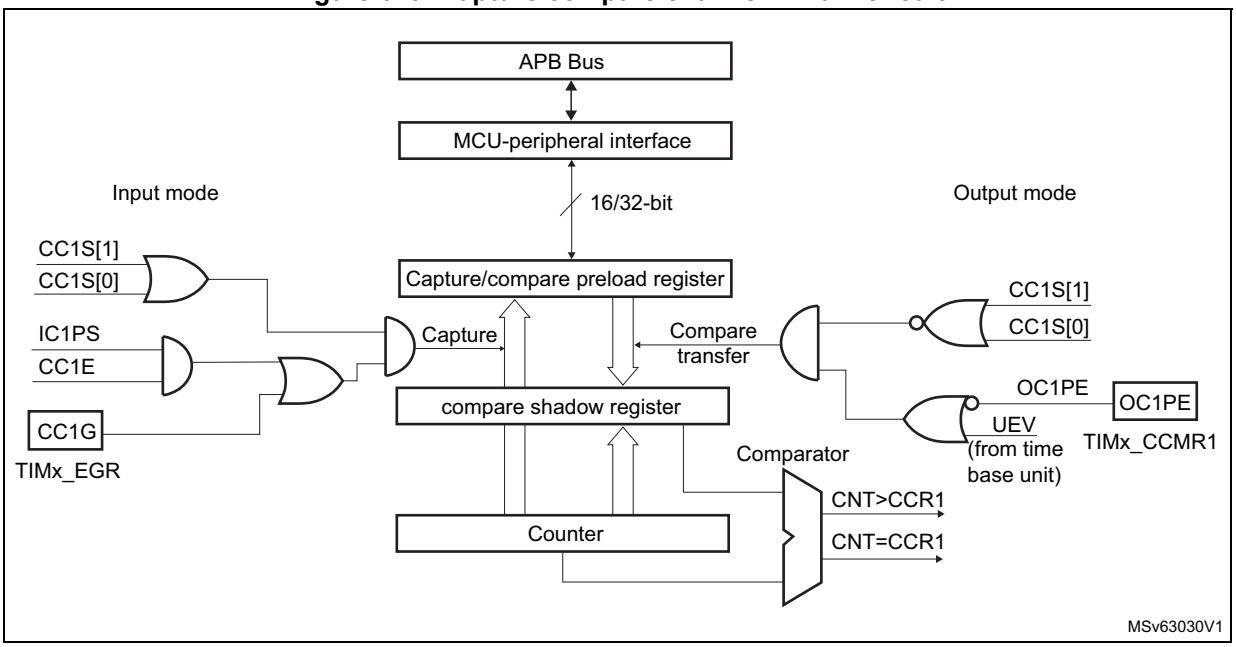


Figure 680. Output stage of capture/compare channel (channel 1)

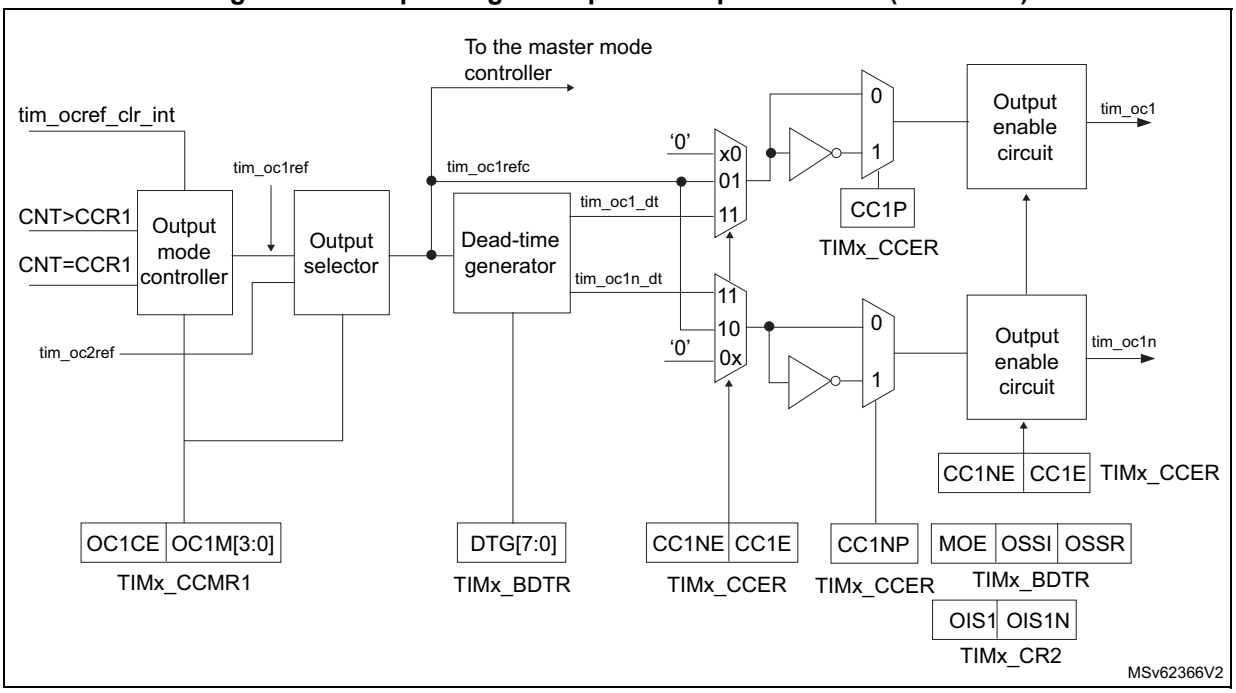
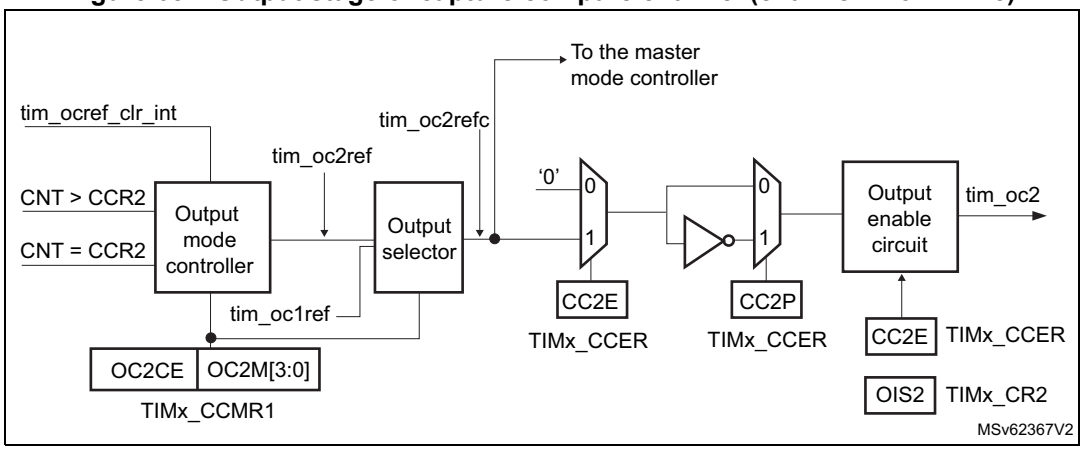


Figure 681. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

57.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

57.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

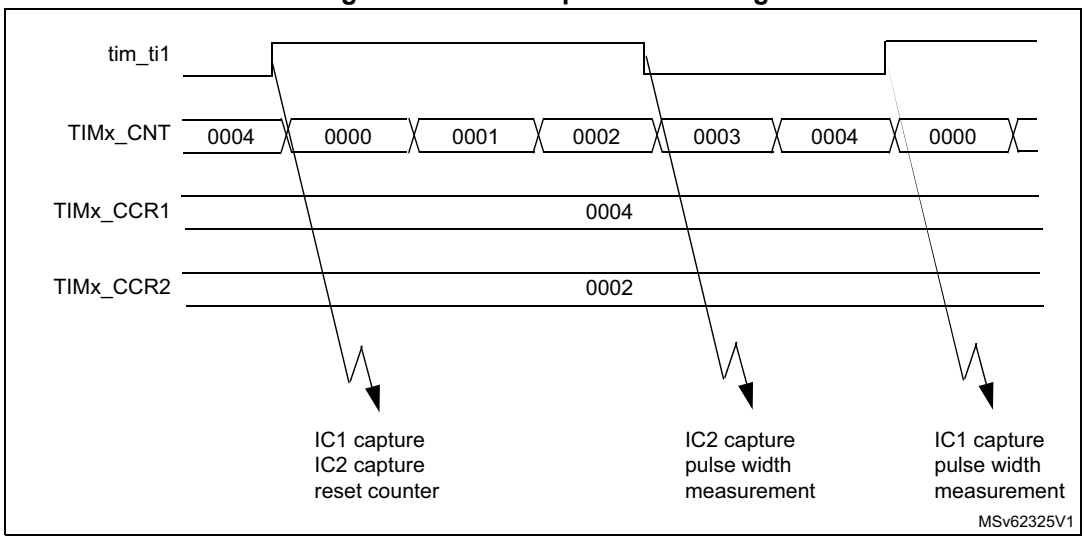
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 682. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

57.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

57.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

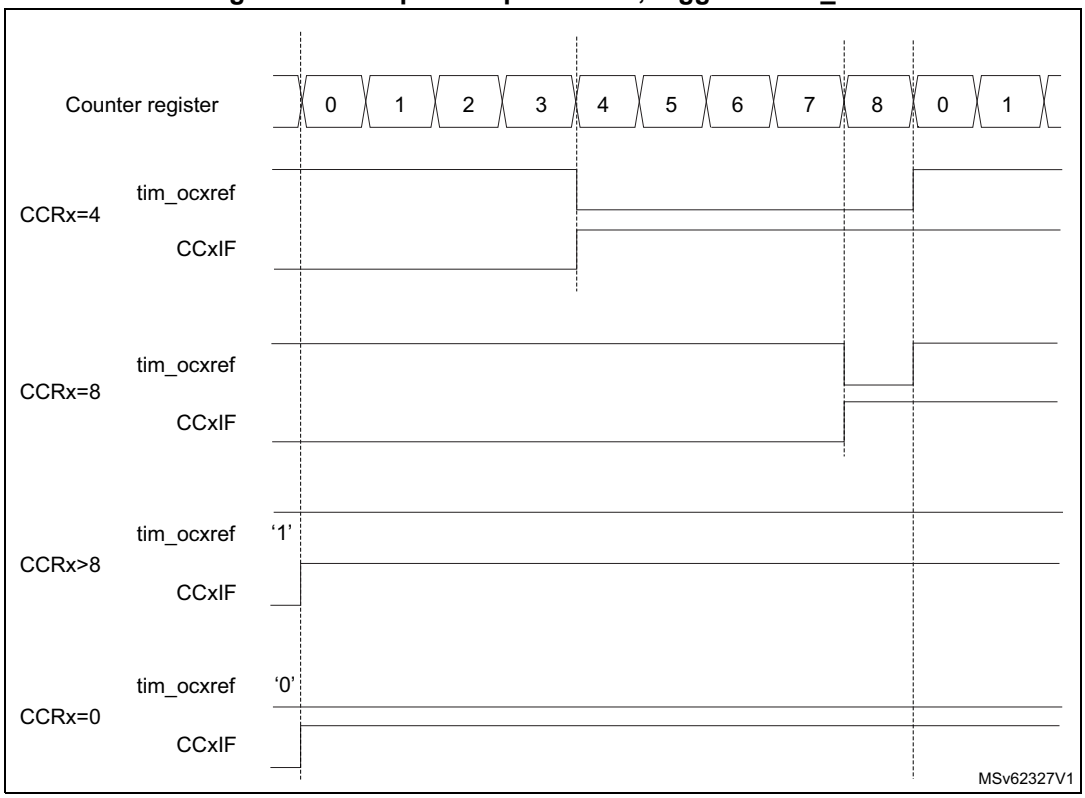
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 683*.

Figure 683. Output compare mode, toggle on tim_oc1



57.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

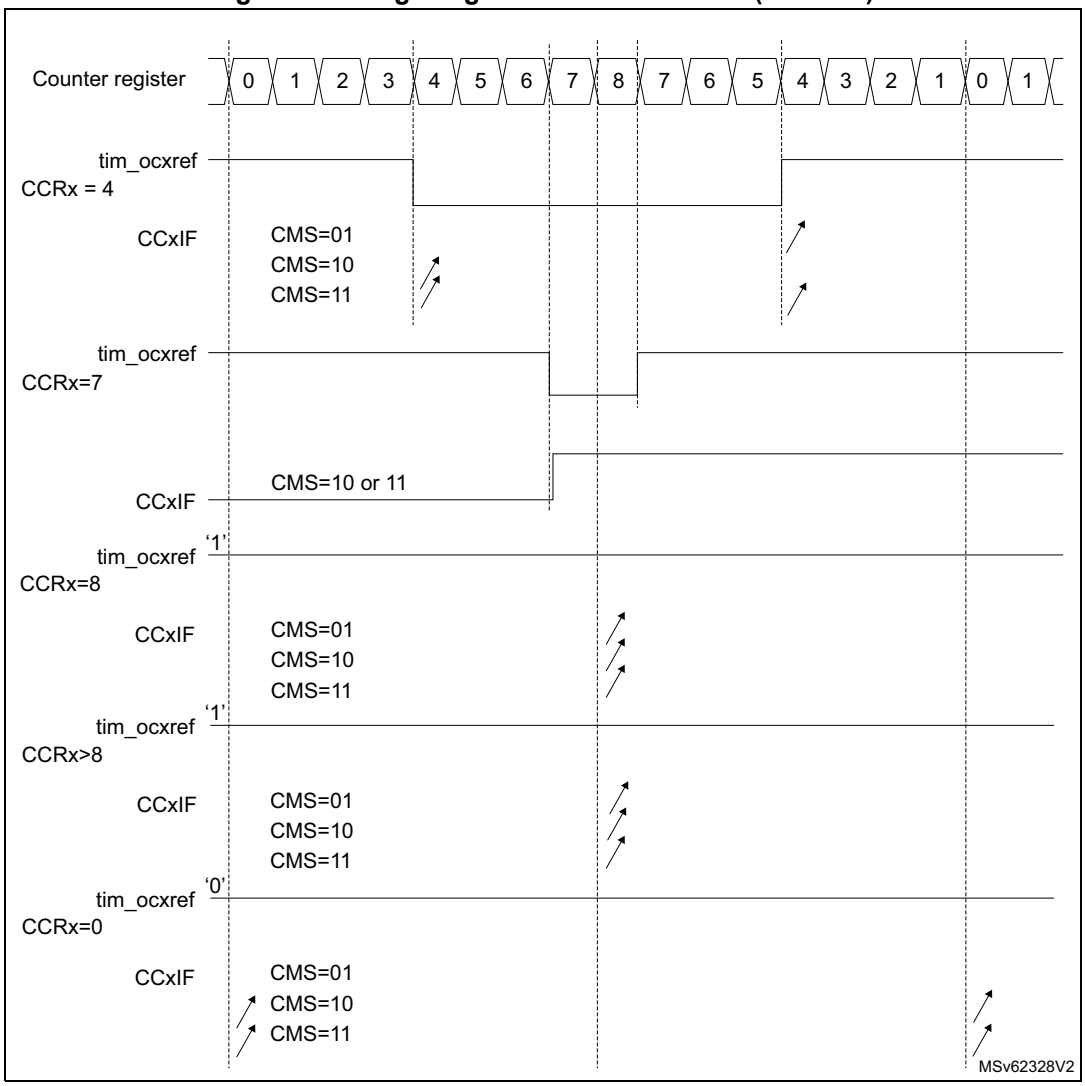
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in

TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 684* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 684. Edge-aligned PWM waveforms (ARR = 8)

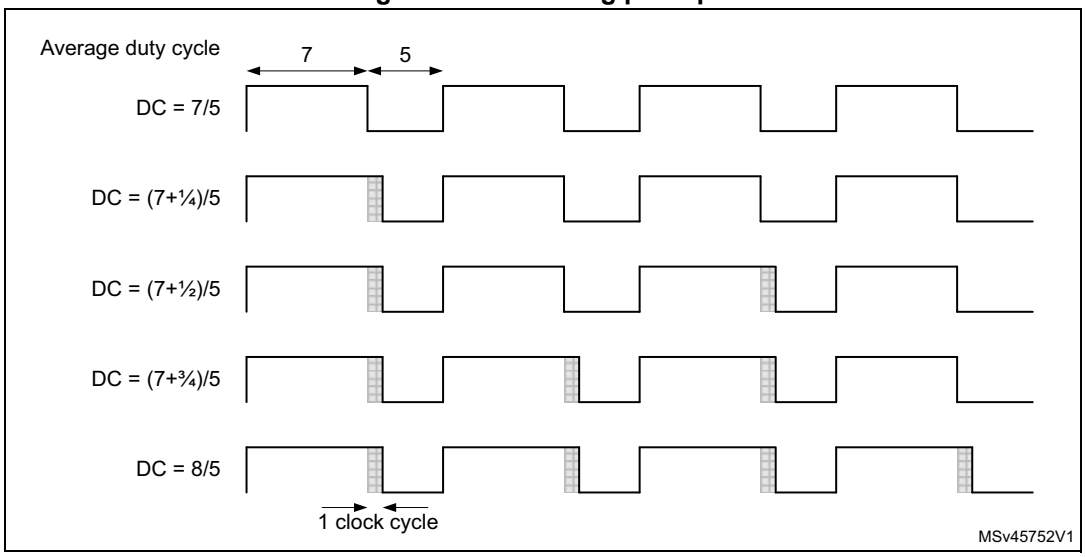


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 685* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 685. Dithering principle

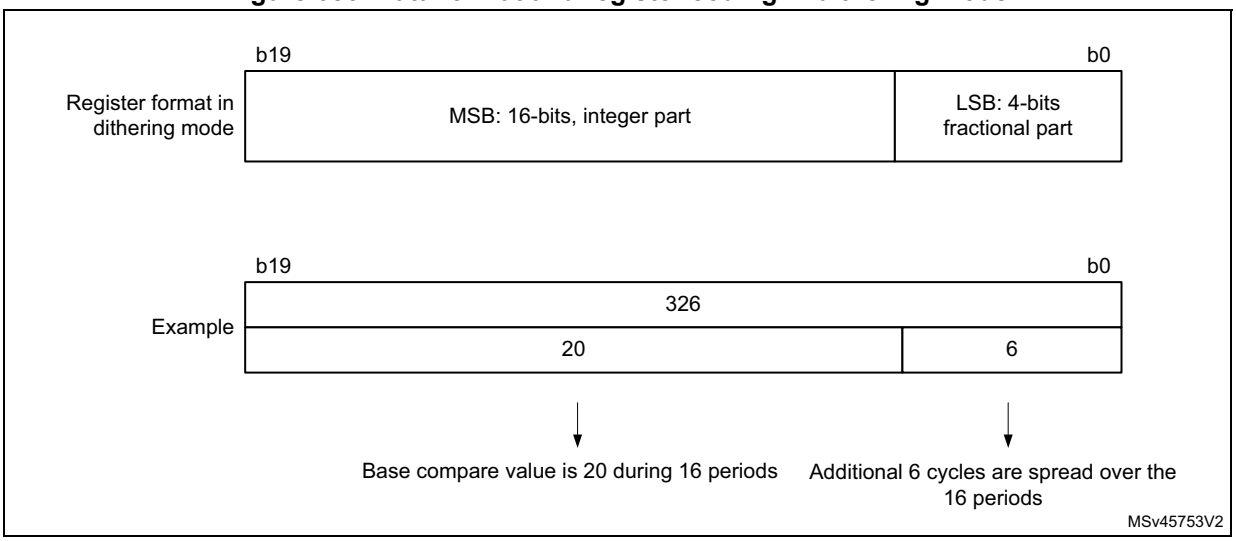


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 686* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 686. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

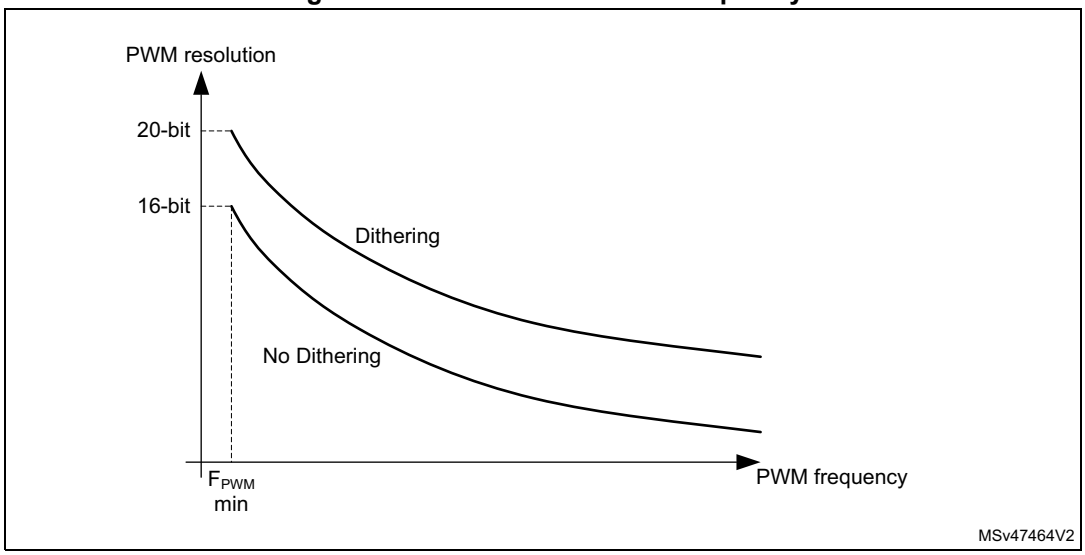
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

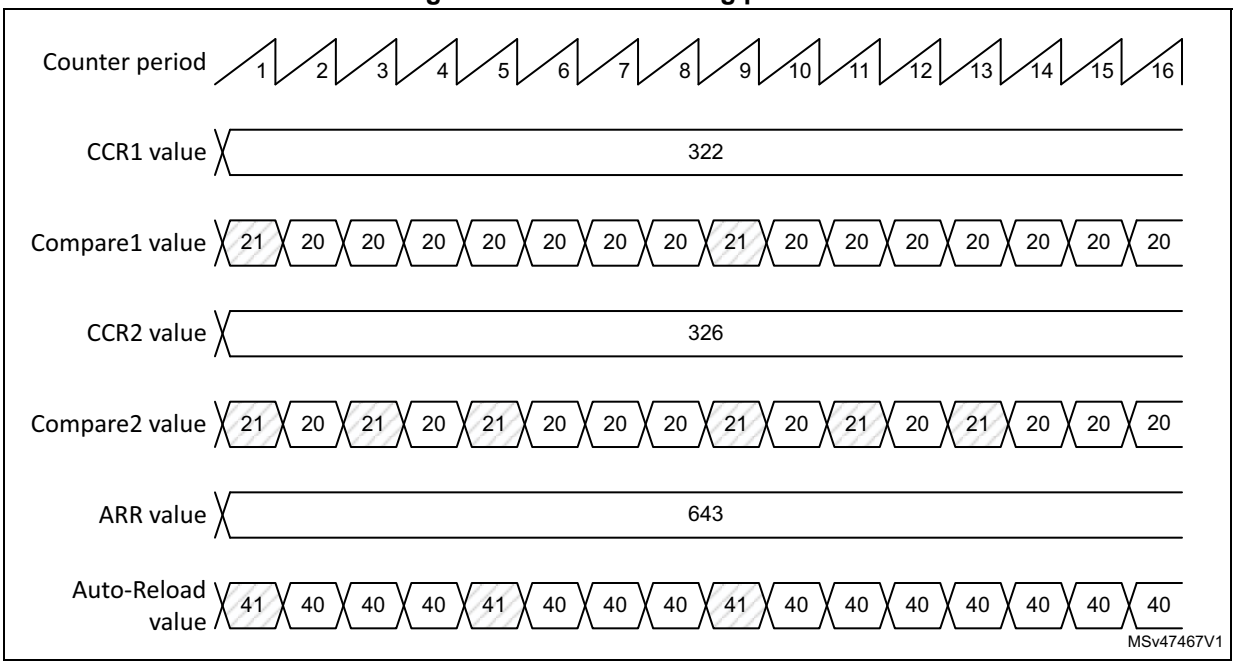
As shown on the *Figure 687* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 687. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 688* below.

Figure 688. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 583* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 583. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 583. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

57.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

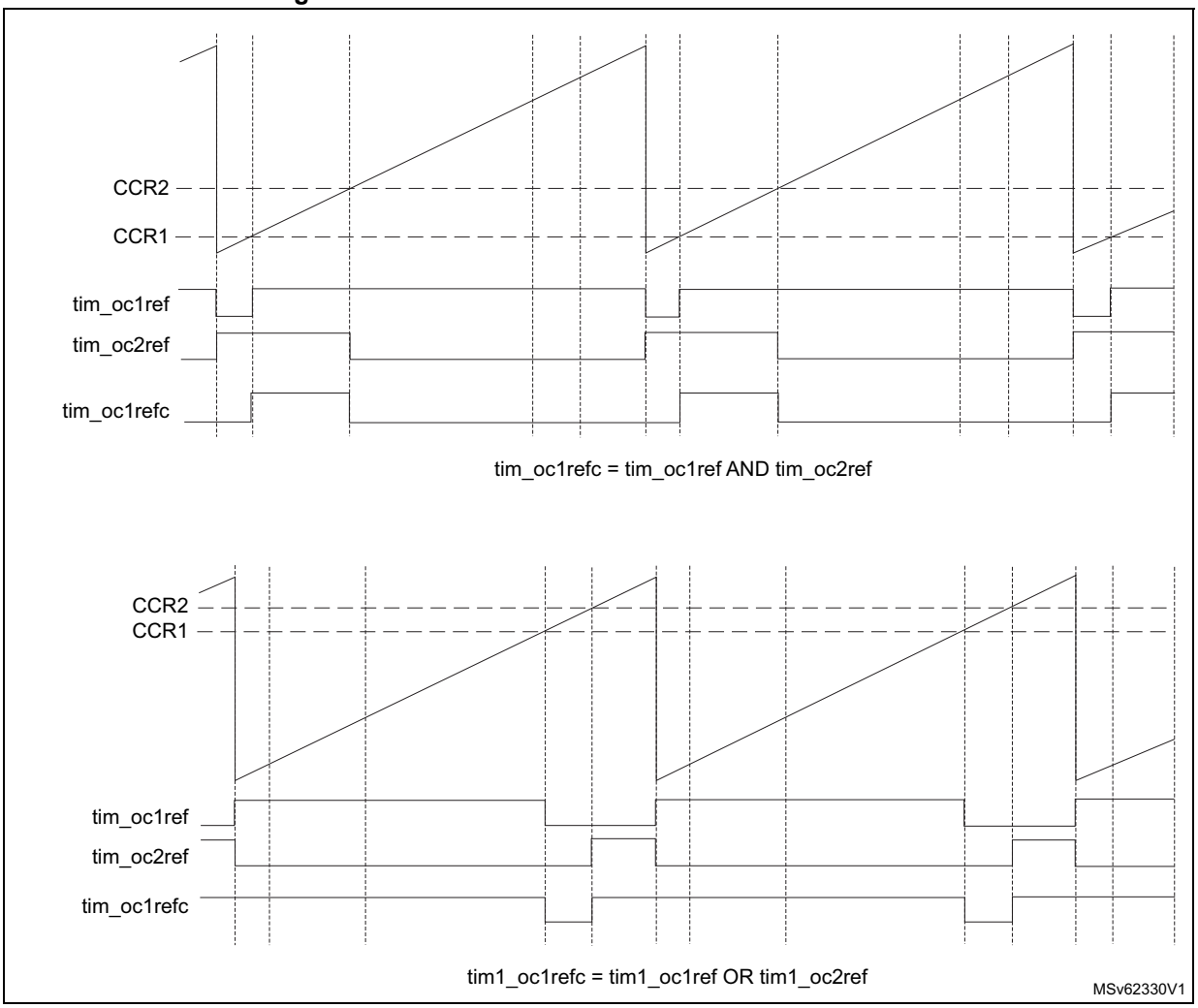
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Table 689 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 689. Combined PWM mode on channel 1 and 2



57.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 590: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

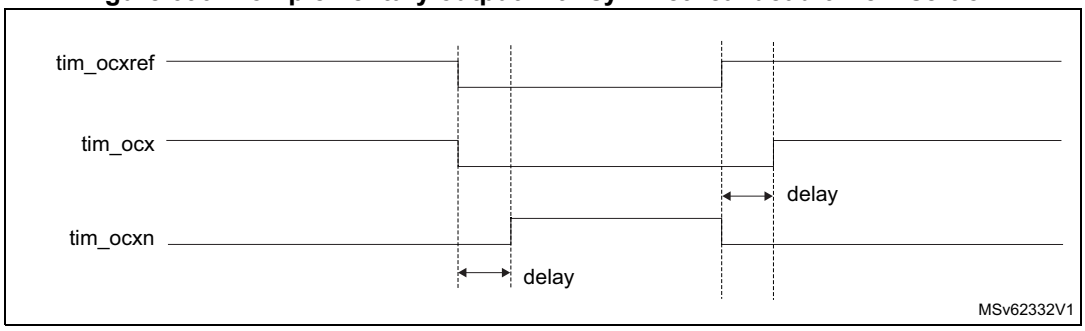
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 690. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 691*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 691. Asymmetrical deadtime

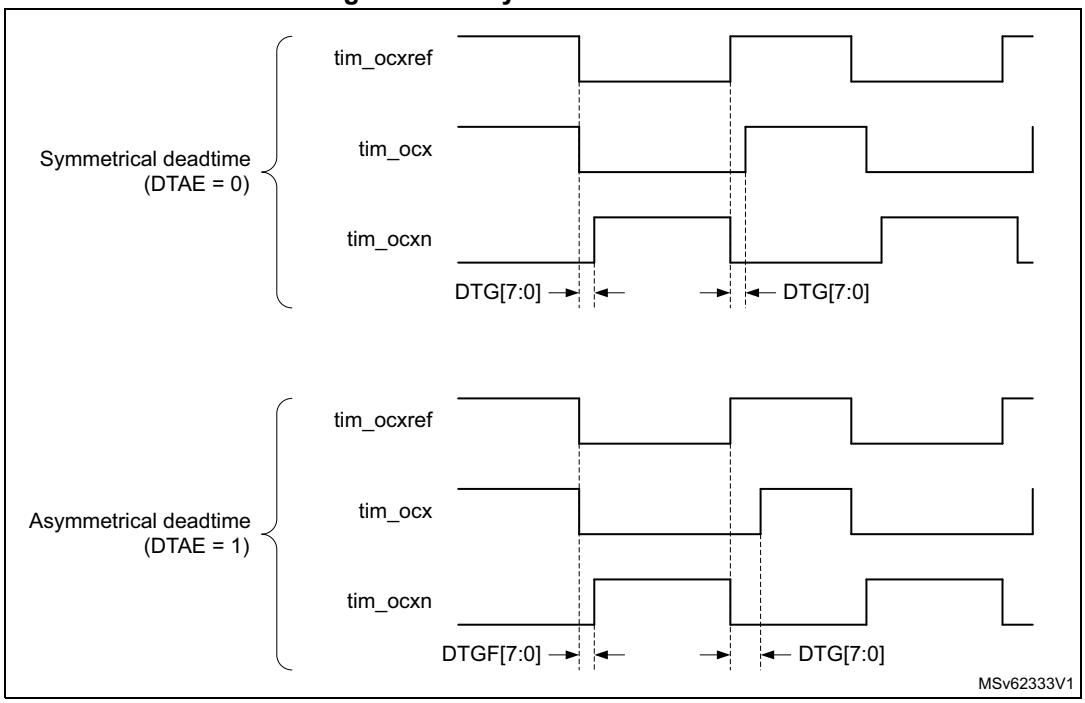


Figure 692. Dead-time waveforms with delay greater than the negative pulse.

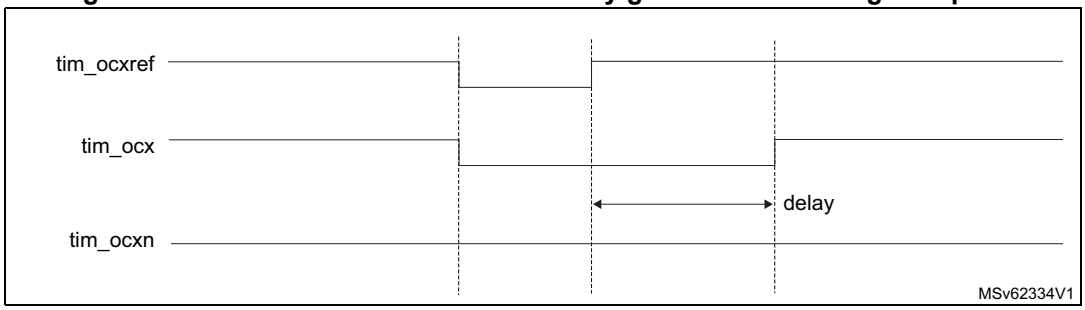
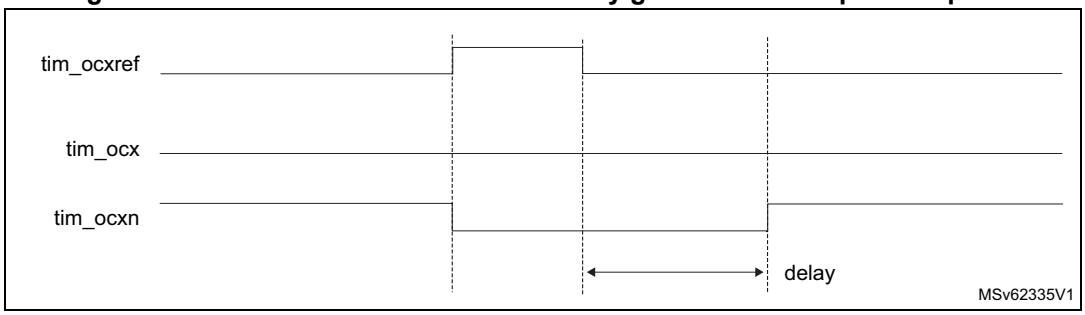


Figure 693. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 57.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

57.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 590: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

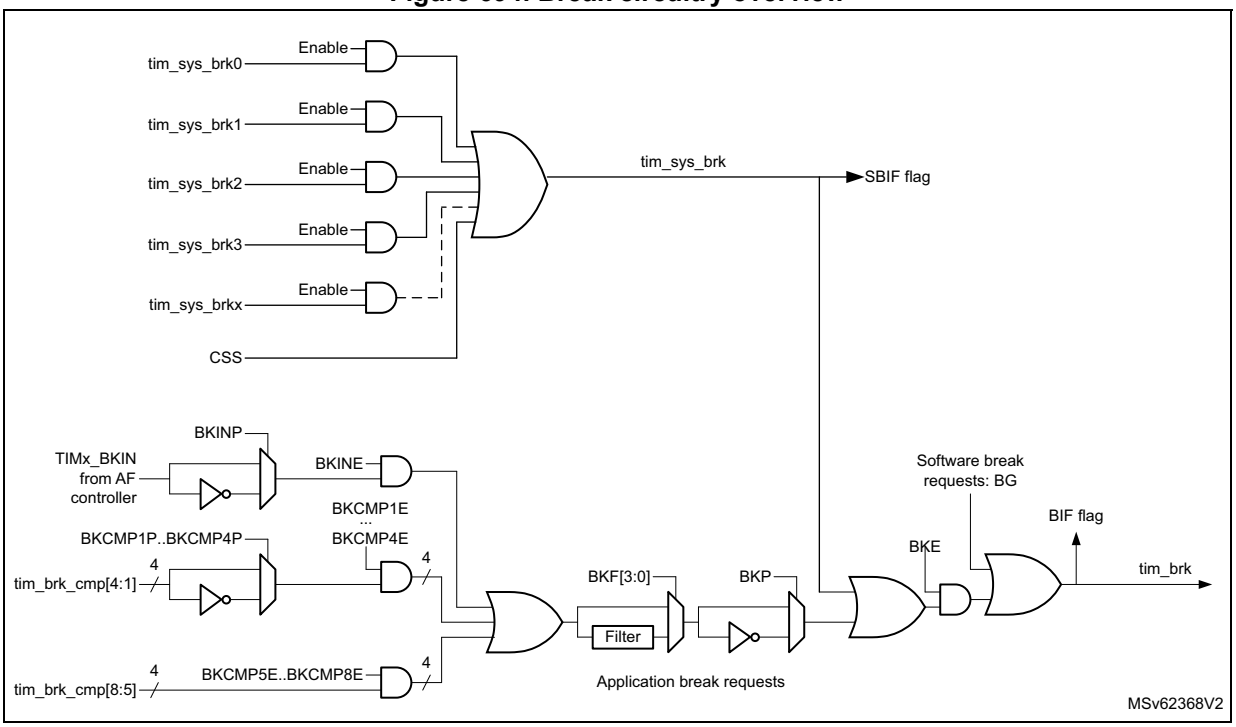
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 694* below.

Figure 694. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

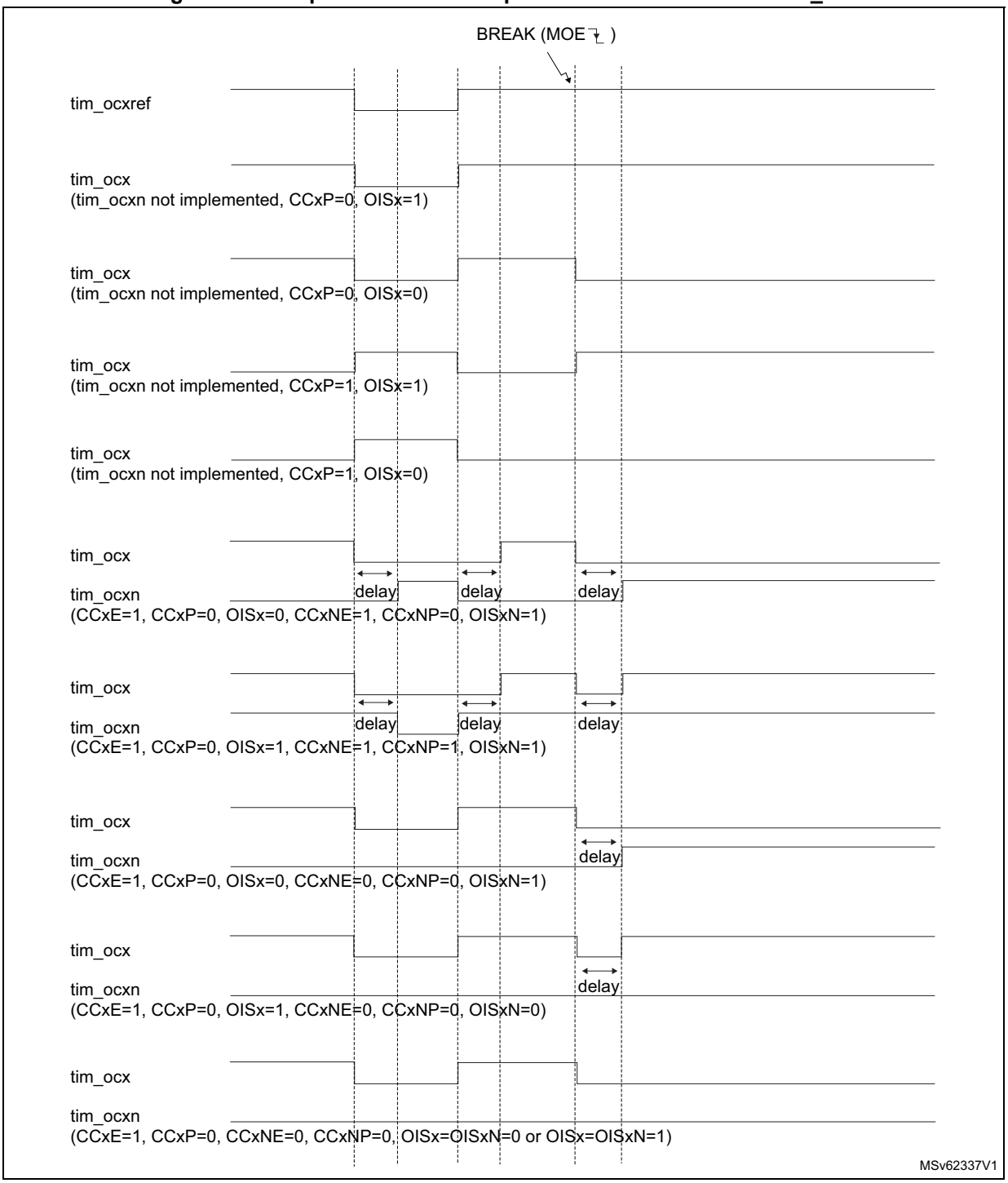
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 57.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 695* shows an example of behavior for the outputs in response to a break.

Figure 695. Output behavior in response to a break event on tim_brk



57.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 696*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMx_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMx_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 584*).

Table 584. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

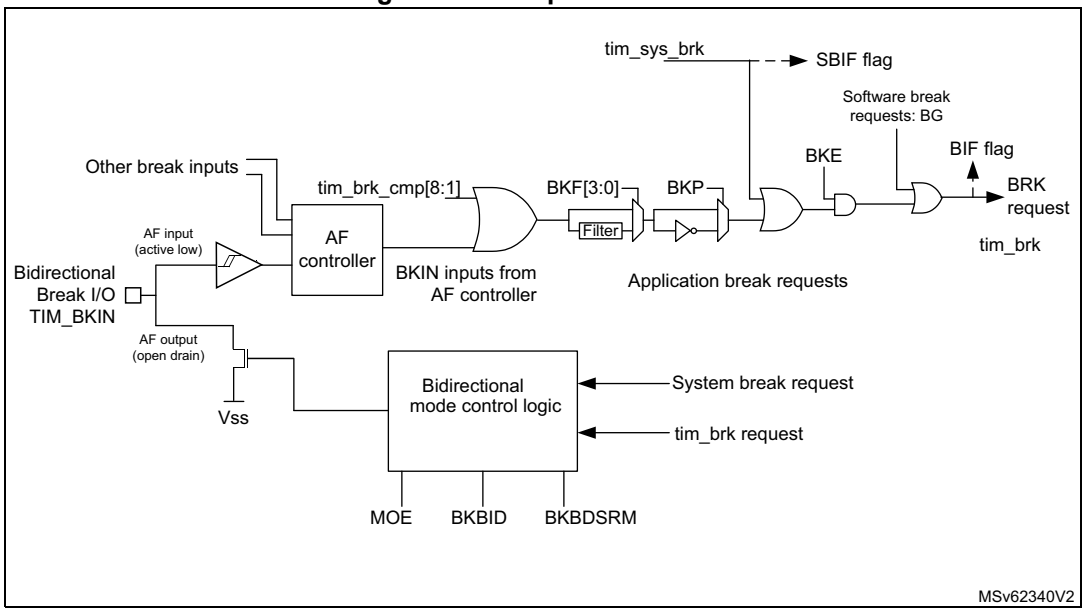
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 696. Output redirection

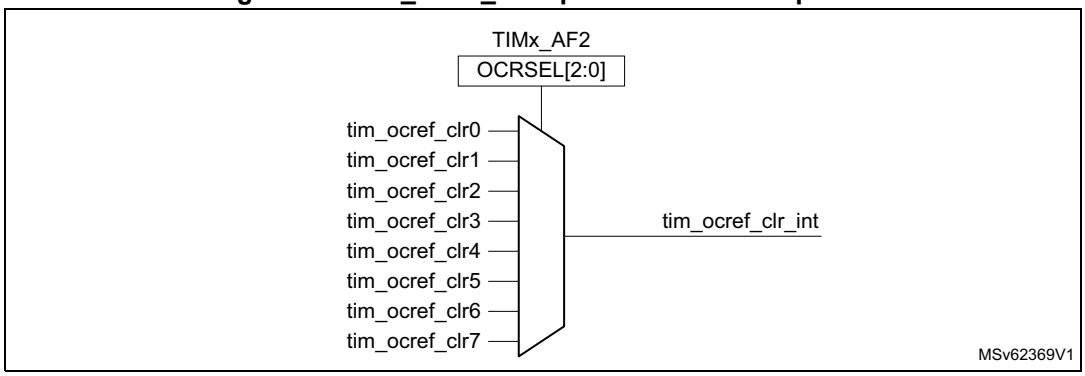


57.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 509* below.

Figure 697. tim_ocref_clr input selection multiplexer



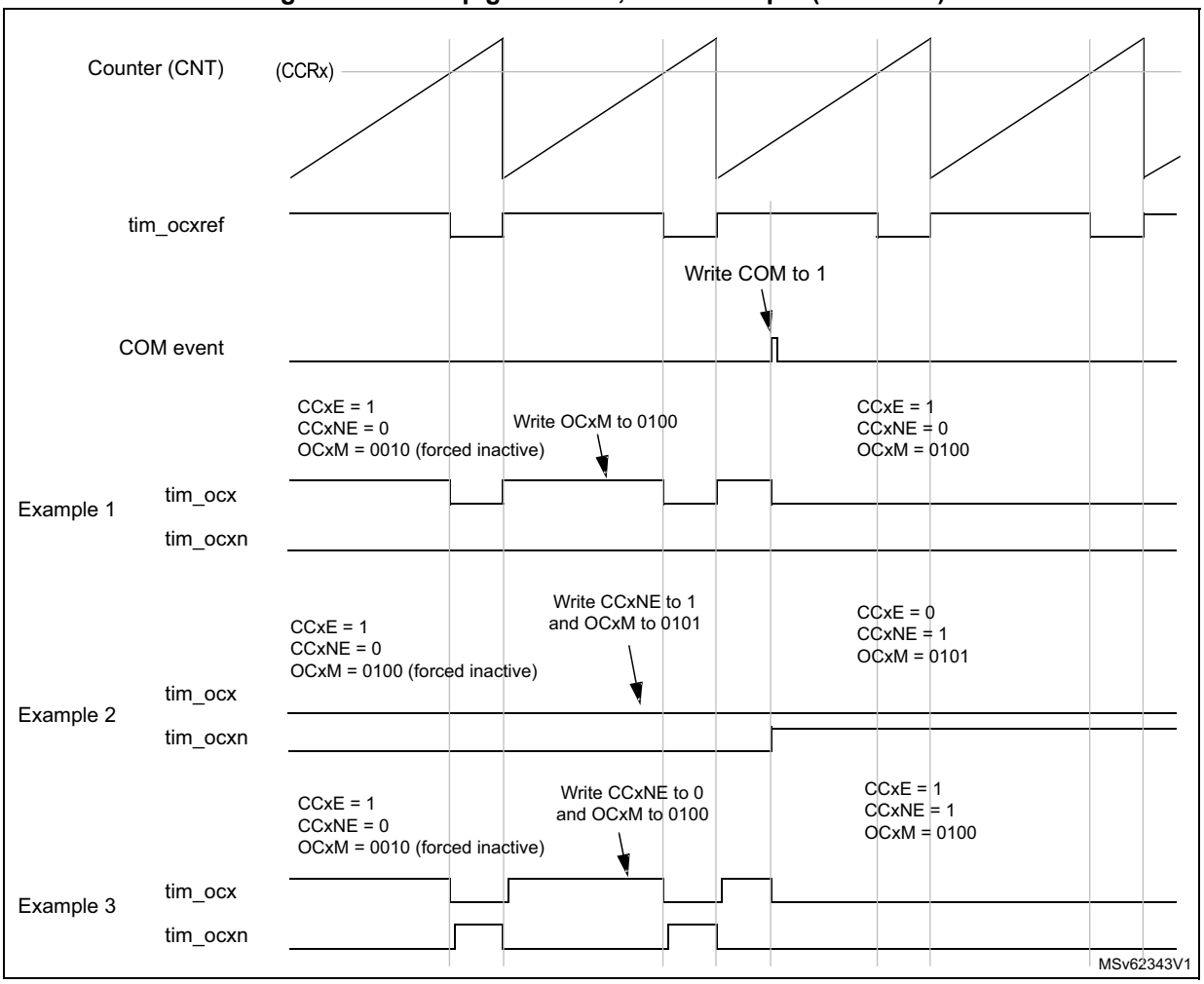
57.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 698* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 698. 6-step generation, COM example (OSSR = 1)



57.4.19 One-pulse mode

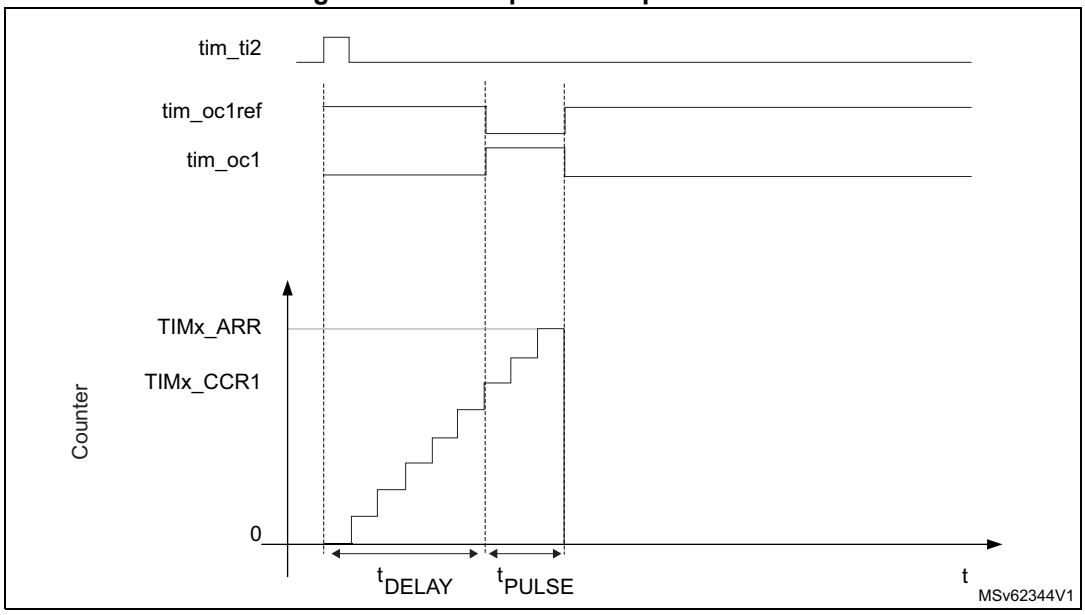
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

Figure 699. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

57.4.20 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 57.4.19*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

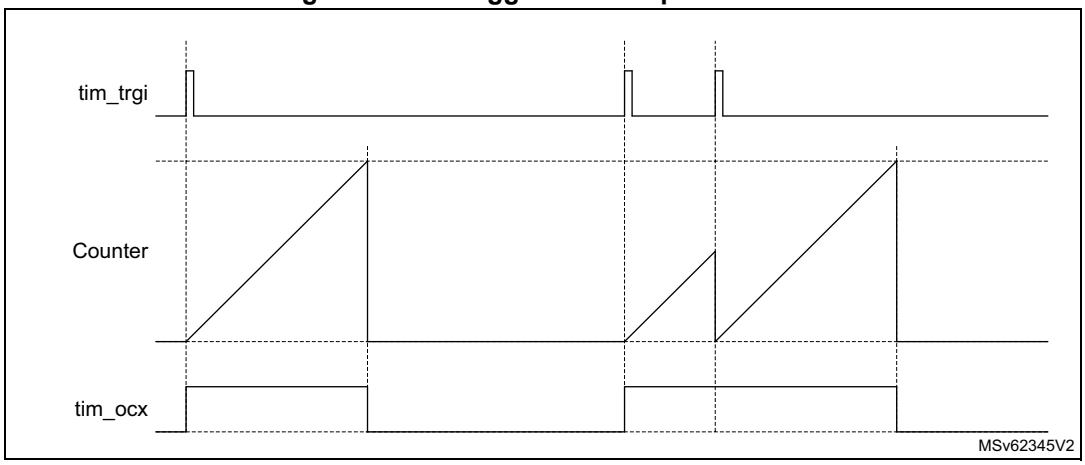
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 700. Retriggerable one pulse mode



57.4.21 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

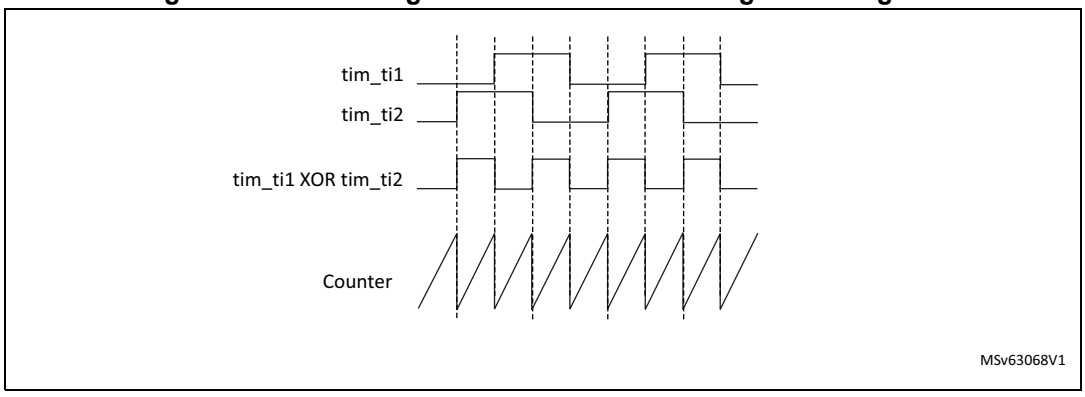
There is no latency between the assertions of the UIF and UIFCPY flags.

57.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 701*.

Figure 701. Measuring time interval between edges on 2 signals



57.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

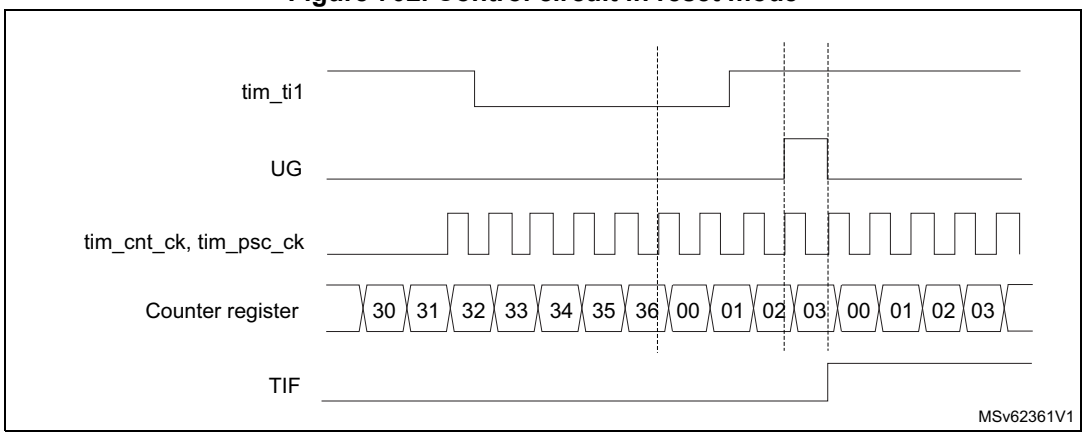
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 702. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

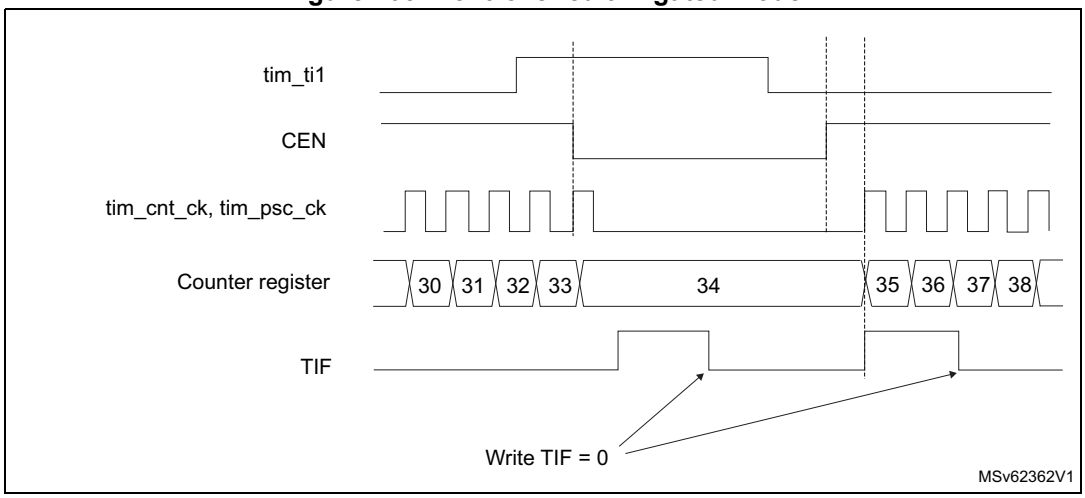
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 703. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

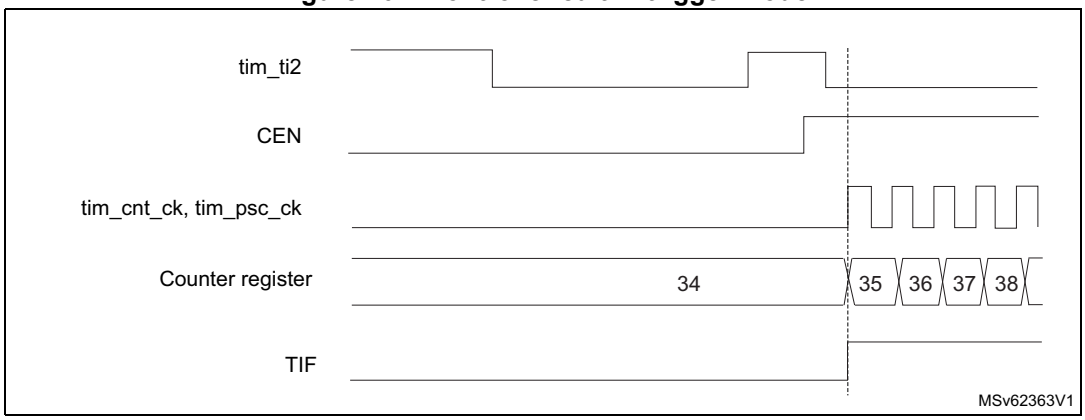
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 704. Control circuit in trigger mode



Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

57.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

57.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

57.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

57.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.4.29 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 57.8.19: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

57.4.30 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 585*.

Table 585. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_com_dma(1)	Commutation (COM)	COMDE
tim_trgi_dma(1)	Trigger	TDE

1. Available for TIM15 only.

57.4.31 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

57.5 TIM15/TIM16/TIM17 low-power modes

Table 586. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

57.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 587*.

Table 587. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

57.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

57.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

57.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2 = 0 when MOE = 0

1: tim_oc2 = 1 when MOE = 0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

57.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
							rw			rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded.

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

57.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS = 0 and UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 57.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

57.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1 A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

57.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

57.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT = TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

57.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 588* for details.

Table 588. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

57.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

57.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

57.7.12 TIM15 autoreload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 57.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

57.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

57.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

57.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

57.7.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 3145*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 3145*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 3145*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS} = 125 ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup
This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.
DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps
Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input
0000: TIM15_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM15_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

57.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

57.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

0011: CC2

0110: COM

0111: Trigger

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

57.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

57.7.23 TIM15 hardware configuration 2 register (TIM15_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width

0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width

0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear

0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

57.7.24 TIM15 hardware configuration 1 register (TIM15_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x1220 1112

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
- 0x0: Down-counter only
- 0x1: Up-counter only
- 0x2: Up/Down counter allowing center-aligned mode
- Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
- 0x0: DMA requests are not possible
- 0x1: Single DMA requests possible
- 0x2: All DMA features implemented
- Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
- 0x0: no break implemented
- 0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
- 0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
- 0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
- Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
- 0x0: External trigger circuit not implemented, ocrefx cleared by ocref_clr input
- 0x1: External trigger circuit implemented, ocrefx cleared by ocref_clr input
- 0x2: External trigger circuit implemented, ocrefx cleared by etrf signal
- 0x3: External trigger circuit implemented, ocrefx cleared by ocref_clr or etrf (1 control bit “OCCS” added in GPT_SMCR)
- Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
- 0x0: repetition counter not implemented
- 0x1: 8-bit repetition counter implemented
- 0x2: 16-bit repetition counter implemented
- Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

57.7.25 TIM15 IP version register (TIM15_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

57.7.26 TIM15 IP identification register (TIM15_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

57.7.27 TIM15 size ID register (TIM15_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

57.7.28 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 589. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIM15_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0

Table 589. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	TIM15_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0				0									0	0	0	0		0	0	0
0x0C	TIM15_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0			0	0	0	0	0	0			0	0	0
0x10	TIM15_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	TIM15_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]	CC1S [1:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIM15_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	TIM15_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM15_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM15_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM15_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM15_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 589. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	TIM15_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIM15_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIM15_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIM15_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	TIM15_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	TIM15_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	TIM15_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIM15_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	TIM15_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_ WIDTH16[3:0]				PSC_ LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIM15_HWCFGR1	CNT_ DIR[3:0]				DMA_ IMPL[3:0]				BK_ IMPL[3:0]				ETR_ IMPL[3:0]				RCR_ IMPL[3:0]				NB_OF_ SM[3:0]				NB_OF_ DT[3:0]				NB_OF_ CC[3:0]			
	Reset value	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0
0x3F4	TIM15_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIM15_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 589. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3FC	TIM15_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

57.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

57.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

57.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

57.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

57.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

57.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

57.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

57.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

57.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 590* for details.

Table 590. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0 tim_oc1n = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n = 0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1 = CC1P	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n = CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

57.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

57.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

57.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 57.4.3: Time-base unit on page 3090* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

57.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

57.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

57.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

57.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

57.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

57.8.20 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

57.8.21 TIMx hardware configuration 2 register (TIMx_HWCFGR2) (x = 16 to 17)

Address offset: 0x3EC

Reset value: 0x0000 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_WIDTH[7:0]								PSC_WIDTH16[3:0]				PSC_LINEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OPTIONREG_OUT[7:0]**: Option register number (0 to 16).

Bits 15:8 **CNT_WIDTH[7:0]**: Counter width

0x08: 8-bits
0x10: 16-bits
0x20: 32-bits
Others: Reserved

Bits 7:4 **PSC_WIDTH16[3:0]**: Prescaler width

0x0: 7 (by power of 2) /8 (linear) -bit counter
0x1: 15 (by power of 2) /16 (linear) -bit counter
Others: Reserved

Bits 3:0 **PSC_LINEAR[3:0]**: Prescaler linear

0x0: prescaler by power of 2 (division by 1, 2, 4, 8...)
0x1: prescaler is linear (division by 1, 2, 3, 4, 5...)
Others: Reserved

57.8.22 TIMx hardware configuration 1 register (TIMx_HWCFGR1) (x = 16 to 17)

Address offset: 0x3F0

Reset value: 0x1220 1011

This read register identifies the hardware configuration parameters of the instantiated IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CNT_DIR[3:0]**: Counter direction
- 0x0: Down-counter only
- 0x1: Up-counter only
- 0x2: Up/Down counter allowing center-aligned mode
- Others: Reserved

Bits 27:24 **DMA_IMPL[3:0]**: DMA features implementation
- 0x0: DMA requests are not possible
- 0x1: Single DMA requests possible
- 0x2: All DMA features implemented
- Others: Reserved

Bits 23:20 **BK_IMPL[3:0]**: Break circuit implementation
- 0x0: no break implemented
- 0x1: break1 implemented without filter (tim_brk + tim_sys_brk connected)
- 0x2: break1 implemented with filter (tim_brk + tim_sys_brk connected)
- 0x3: break 1 and 2 implemented (tim_brk + tim_sys_brk + brk2)
- Others: Reserved

Bits 19:16 **ETR_IMPL[3:0]**: External trigger implementation
- 0x0: External trigger circuit not implemented, ocrefx cleared by ocref_clr input
- 0x1: External trigger circuit implemented, ocrefx cleared by ocref_clr input
- 0x2: External trigger circuit implemented, ocrefx cleared by etrf signal
- 0x3: External trigger circuit implemented, ocrefx cleared by ocref_clr or etrf (1 control bit “OCCS” added in GPT_SMCR)
- Others: Reserved

Bits 15:12 **RCR_IMPL[3:0]**: Repetition counter implementation
- 0x0: repetition counter not implemented
- 0x1: 8-bit repetition counter implemented
- 0x2: 16-bit repetition counter implemented
- Others: Reserved

Bits 11:8 **NB_OF_SM[3:0]**: Number of synchronization module
0x0: synchronization module not implemented
0x1: 1 synchronization module implemented
Others: Reserved

Bits 7:4 **NB_OF_DT[3:0]**: Number of complementary outputs and dead-time generators
0x0
0x1
0x2
0x3
0x4
Others: Reserved

Bits 3:0 **NB_OF_CC[3:0]**: Number of capture / compare channel
0x0: no capture/compare channel implemented
0x1: only CC1 channel is implemented
0x2: CC1 and CC2 channels are implemented
0x3: CC1, CC2 and CC3 channels are implemented
0x4: CC1, CC2, CC3 and CC4 channels are implemented
0x5: CC1, CC2, CC3, CC4 and CC5 channels are implemented
0x6: CC1, CC2, CC3, CC4, CC5 and CC6 channels are implemented
Others: Reserved

57.8.23 TIMx IP version register (TIMx_VERR)(x = 16 to 17)

Address offset: 0x3F4

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision number.
0x4

Bits 3:0 **MINREV[3:0]**: Minor revision number.
0x3

57.8.24 TIMx IP identification register (TIMx_IPIDR)(x = 16 to 17)

Address offset: 0x3F8

Reset value: 0x0012 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code.
0x00120002

57.8.25 TIMx size ID register (TIMx_SIDR)(x = 16 to 17)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SID[31:8]**: Fixed code.
0xA3C5DD

Bits 7:0 **SID[7:0]**: Offset in KB of the ID register base address from the IP base address.
0x1: 1KB offset

57.8.26 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 591. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE		COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 591. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	TIMx_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTIONREG_OUT[7:0]								CNT_WIDTH[3:0]								PSC_WIDTH16 [3:0]				PSC_LINEAR[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0x3F0	TIMx_HWCFGR1	CNT_DIR[3:0]				DMA_IMPL[3:0]				BK_IMPL[3:0]				ETR_IMPL[3:0]				RCR_IMPL[3:0]				NB_OF_SM[3:0]				NB_OF_DT[3:0]				NB_OF_CC[3:0]			
	Reset value	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1

Table 591. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F4	TIMx_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0x3F8	TIMx_IPIDR	ID[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	TIMx_SIDR	SID[31:16]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

58 Low-power timer (LPTIM)

58.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

58.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

58.3 LPTIM implementation

Table 592 describes LPTIM implementation on STM32MP21xx devices. The full set of features is implemented in LPTIM1 and LPTIM2 while LPTIM3, LPTIM4 and LPTIM5 support a smaller set of features.

Table 592. STM32MP21xx LPTIM features

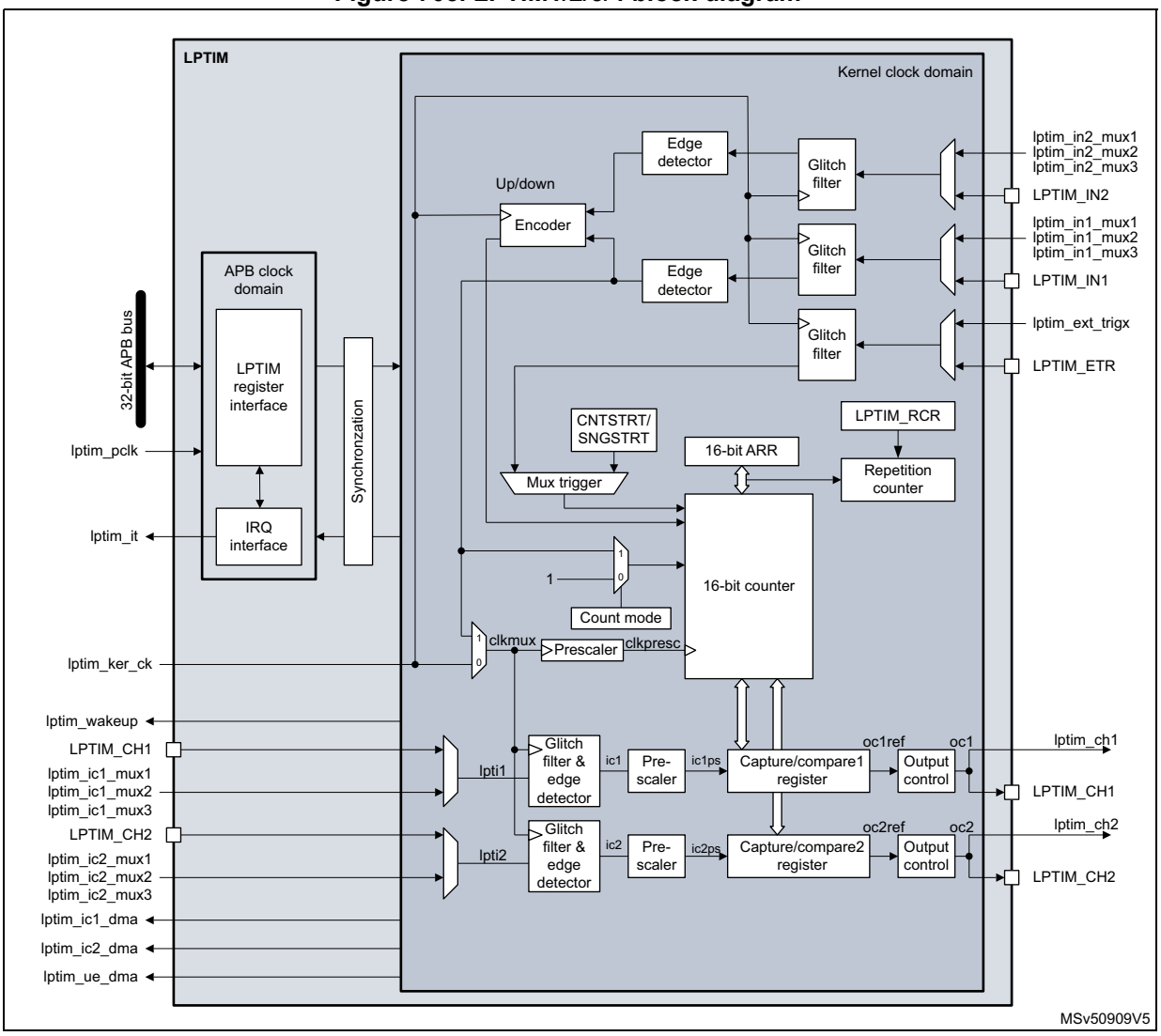
LPTIM modes/features(1)	LPTIM1	LPTIM2	LPTIM3(2)	LPTIM4	LPTIM5
Encoder mode	X	X	-	-	-
PWM mode	X	X	X	X	X
Input Capture	X	X	X	X	-
Number of channels	2	2	2	2	0
Number of DMA requests	3	3	3	3	0
Wake-up in Stop1/2 modes	X	X	X	X	X
Autonomous in LP-Stop1/2, LPLV-Stop1/2, Standby modes	X	X	X	X	-

1. X = supported.
2. LPTIM3_IN1 and LPTIM3_CH2 are only available on same GPIO and so cannot be used together.

58.4 LPTIM functional description

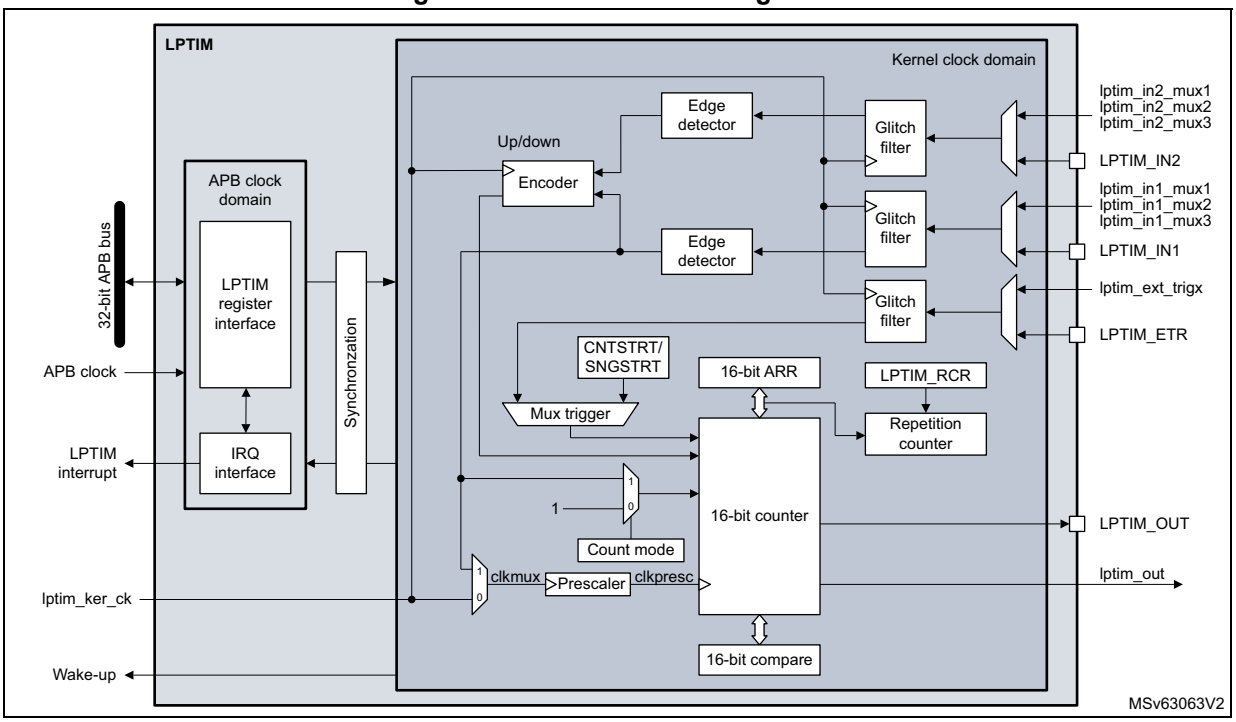
58.4.1 LPTIM block diagram

Figure 705. LPTIM1/2/3/4 block diagram(1)



1. Some I/Os may not be available, refer to *Section 58.4.2: LPTIM pins and internal signals*.

Figure 706. LPTIM5 block diagram(1)



1. Some I/Os may not be available, refer to *Section 58.4.2: LPTIM pins and internal signals*.

58.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 593. LPTIM1/2/3/4 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2(1)	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

1. LPTIM3/4 have only the input IN1, no input IN2.

Table 594. LPTIM5 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM output GPIO pin

Table 595. LPTIM1/2/3/4 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1(1)	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2(1)	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3(1)	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

1. LPTIM3/4/5 have only the input 1 (no input 2).

Table 596. LPTIM5 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

58.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 597. LPTIM1/2/3/4/5 external trigger connections

TRIGSEL	External trigger				
	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
lptim_ext_trig0	GPIO				
lptim_ext_trig1	rtc_alra_trg				
lptim_ext_trig2	rtc_alrb_trg				
lptim_ext_trig3	tamp_trg1				
lptim_ext_trig4	tamp_trg2				
lptim_ext_trig5	HSI/1024	tamp_trg3			
lptim_ext_trig6	sai2_fsa	sai4_fsa	lptim4_ch1	lptim3_ch1	lptim3_ch1
lptim_ext_trig7	sai2_fsb	sai4_fsb	lptim5_out	lptim5_out	lptim4_ch1

Table 598. LPTIM1/2/3/4/5 input 1 connections

lptim_in1_mux	LPTIM1/2/3/4/5 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	HSE
lptim_in1_mux2	Reserved
lptim_in1_mux3	Reserved

Table 599. LPTIM1/2 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	Reserved
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 600. LPTIM1/2/3/4 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to	LPTIM2 input capture 1 connected to	LPTIM3 input capture 1 connected to	LPTIM4 input capture 1 connected to
lptim_ic1_mux0	GPIO	GPIO	GPIO	GPIO
lptim_ic1_mux1	I3C1_IBIACK	I3C1_IBIACK	MSI	LSI
lptim_ic1_mux2	I3C2_IBIACK	I3C2_IBIACK	I3C2_IBIACK	LSE
lptim_ic1_mux3	I3C3_IBIACK	I3C3_IBIACK	I3C3_IBIACK	Reserved

Table 601. LPTIM1/2/3 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to	LPTIM2 input capture 2 connected to	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	GPIO	GPIO	GPIO
lptim_ic2_mux1	HSI/1024	Reserved	Reserved
lptim_ic2_mux2	MSI	Reserved	Reserved
lptim_ic2_mux3	Reserved	Reserved	Reserved

58.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

58.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

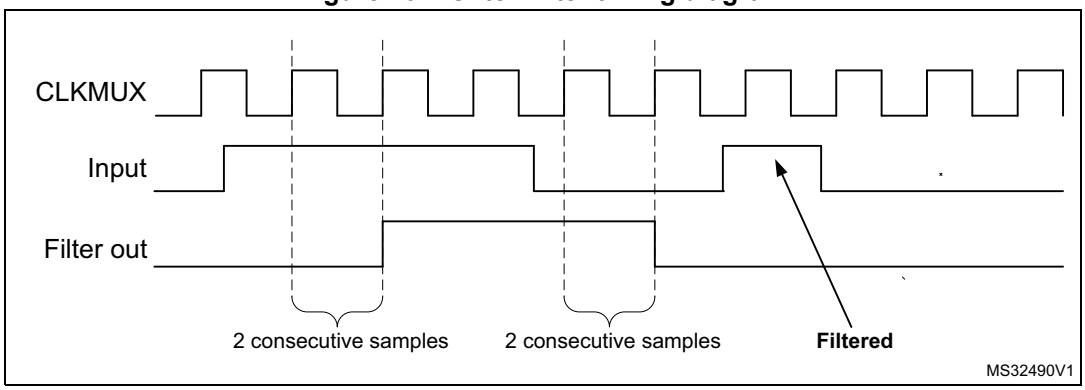
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 707* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 707. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

58.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 602. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

58.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible

values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.

- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

58.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

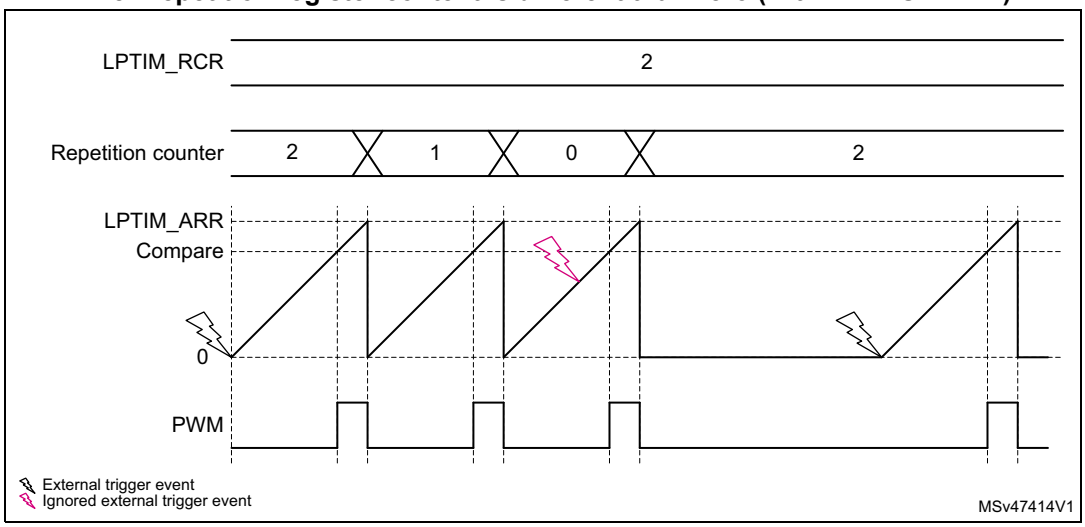
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 708*.

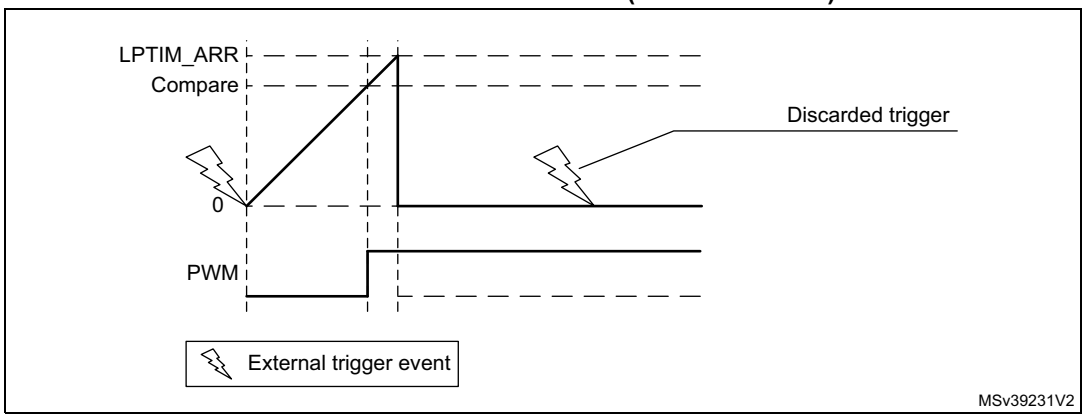
Figure 708. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 709*.

Figure 709. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

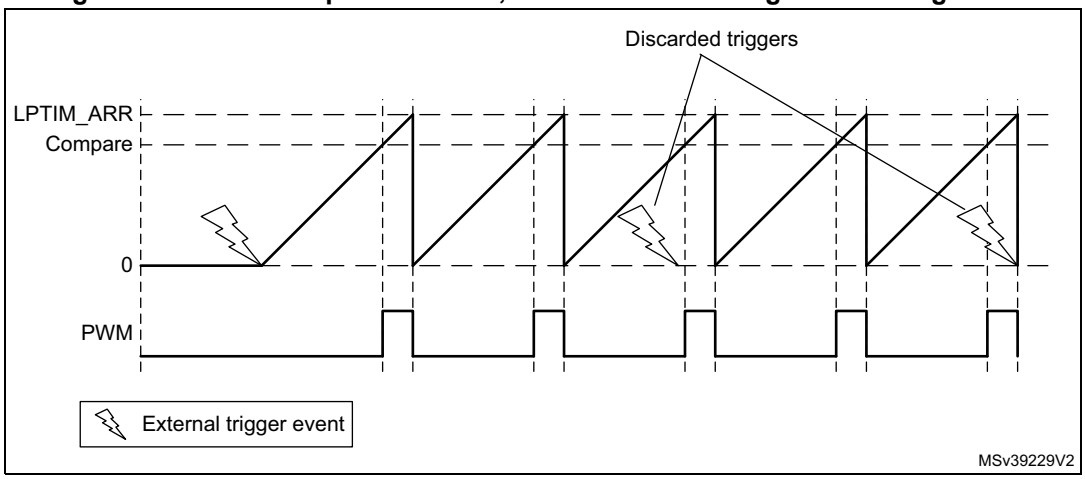
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 710*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 710. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

58.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

58.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 58.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

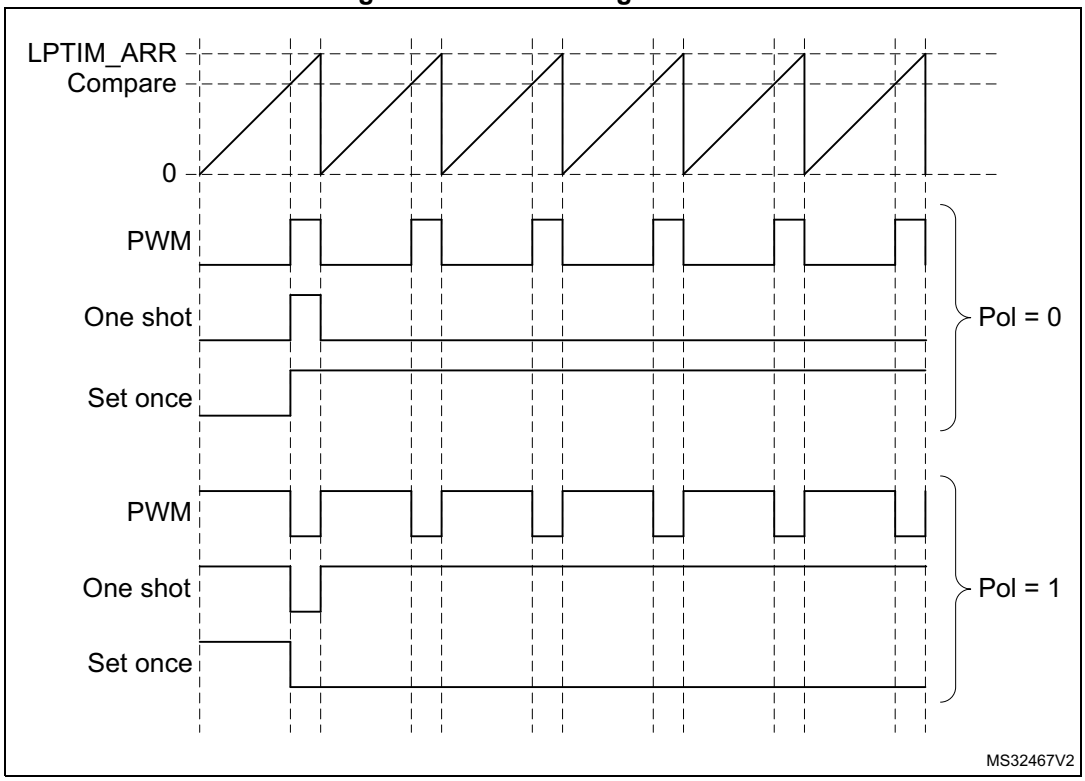
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 711* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

Figure 711. Waveform generation



58.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

58.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as

system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

58.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

58.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

58.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

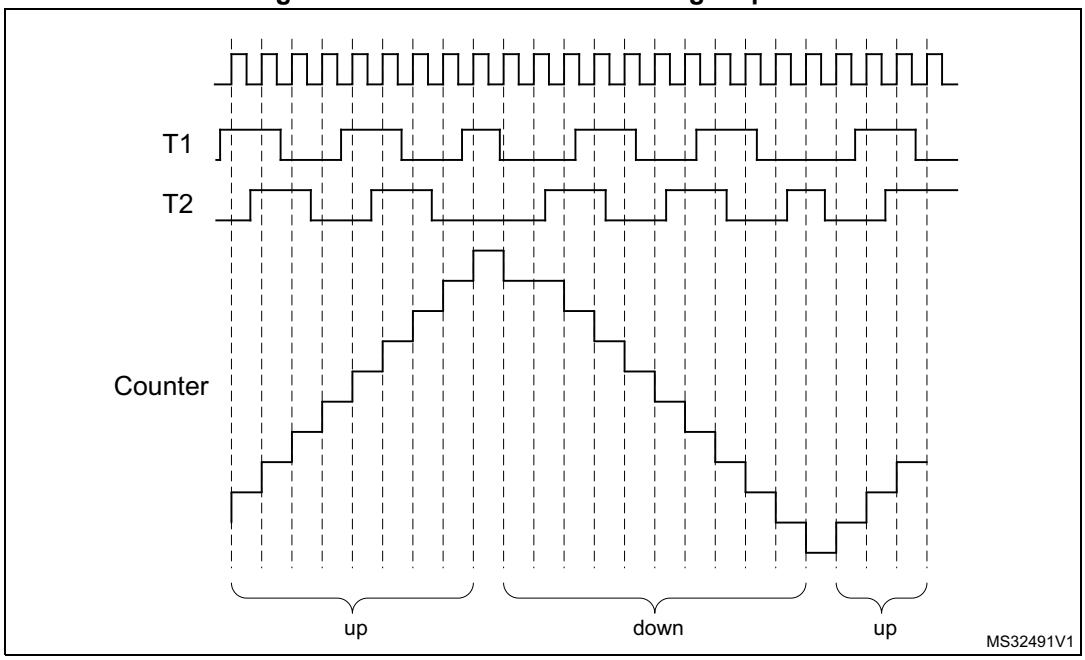
Table 603. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 712. Encoder mode counting sequence



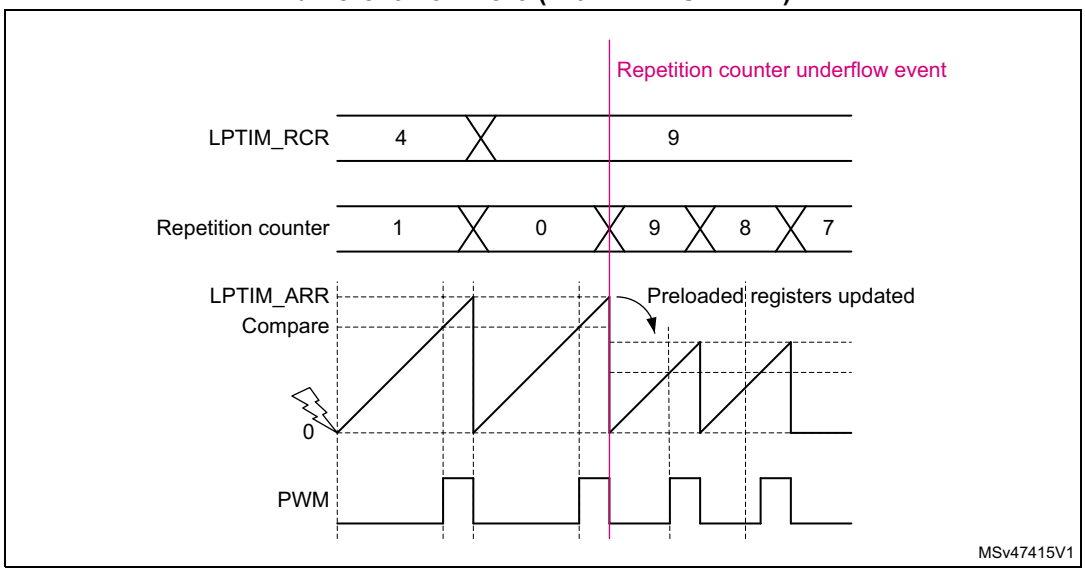
58.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 713*.

Figure 713. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 58.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

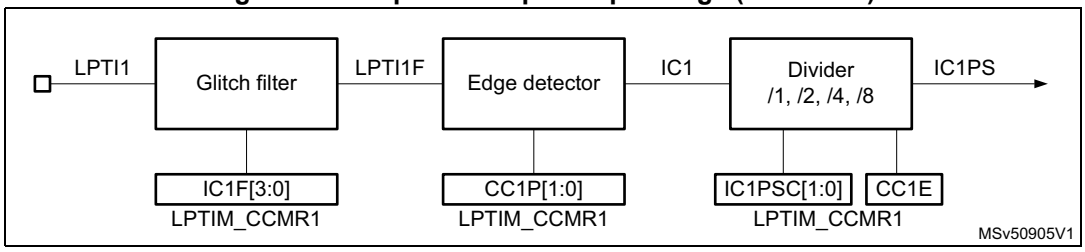
58.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

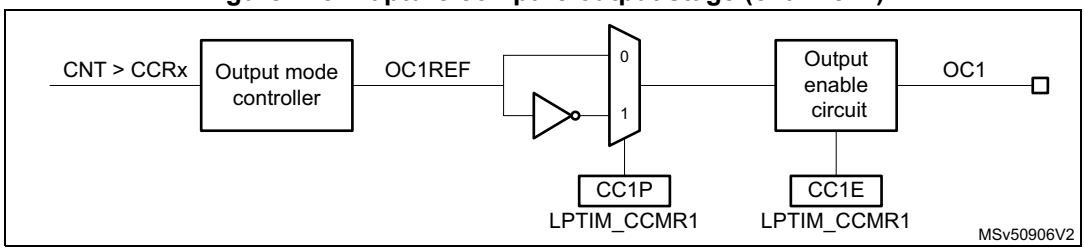
Figure 714. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 715. Capture/compare output stage (channel 1)



58.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 604*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 604*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 604. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 604. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

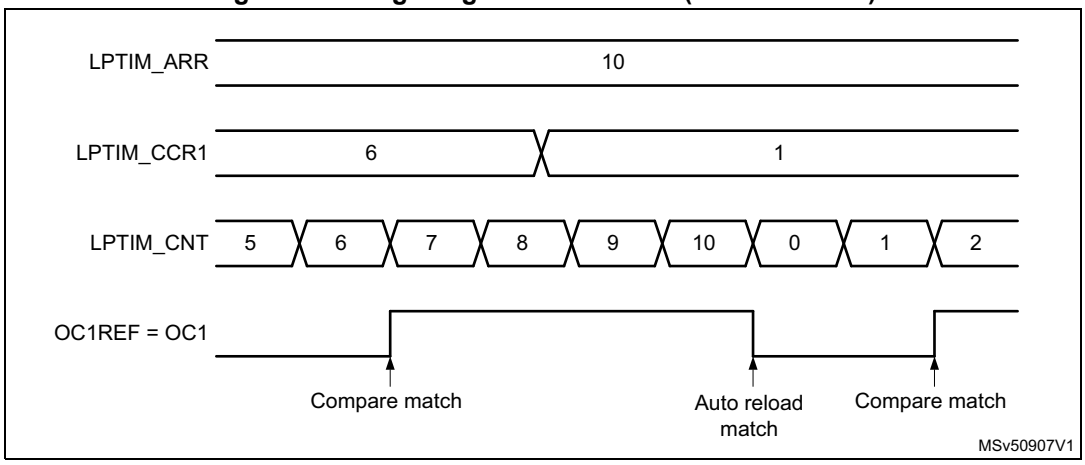
58.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 716 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

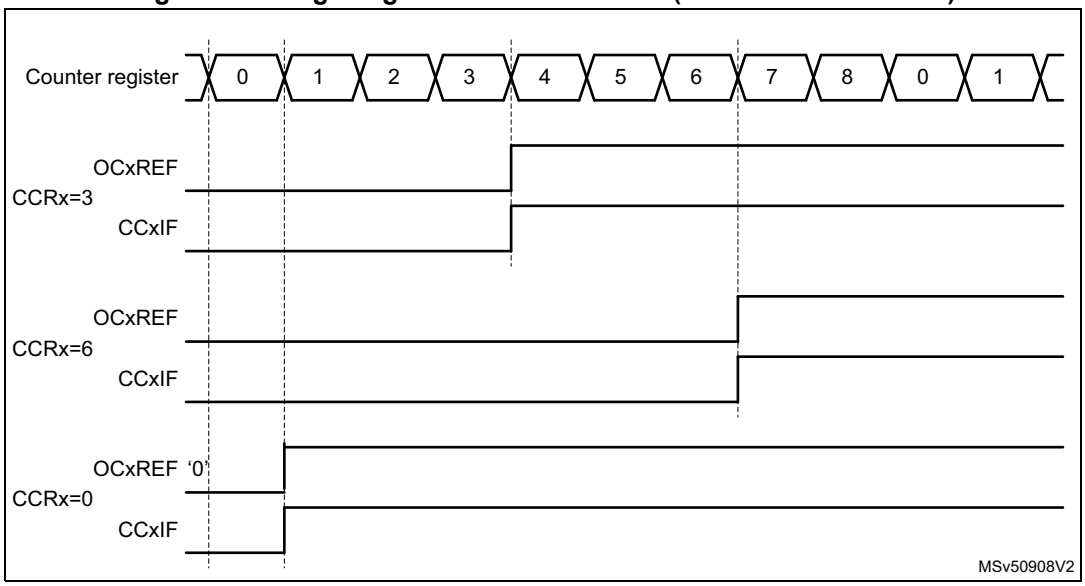
Figure 716. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 717 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 717. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

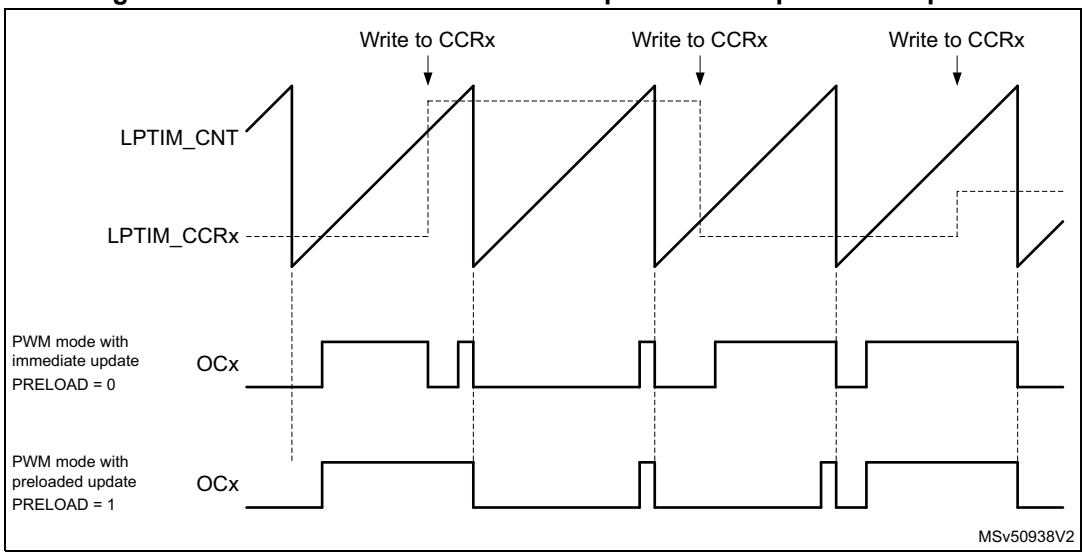
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 718* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 718. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 716 illustrates this behavior.

58.4.20 Autonomous mode

When clocked by oscillators available in this mode (refer to RCC), the LPTIM can operate in autonomous mode, remaining then fully functional in Stop mode where the APB clock is stopped. The APB clock is requested by the peripheral each time a data must be transferred from or to the SRAM. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the LPTIM configuration.

In order to offload the CPU (in Run mode) or to avoid to wake it up when in Stop mode, it is possible to use LPTIM DMA requests to transfer the captured values when in input capture mode or to update LPTIM registers when in PWM mode.

When in Stop mode, the LPTIM counter can be automatically started after the detection of an active edge on one of its external input triggers.

Input capture mode

To operate autonomously in Stop mode, the input capture DMA request must be enabled by setting the CCxDE bit in the LPTIM_DIER register.

Each time a counter value is captured and available in the LPTIM_CCRx register, the APB clock is requested by the peripheral and a DMA request is generated. The captured value is then transferred to the SRAM. The CCxIF flag is automatically cleared by hardware once the captured value is read by APB (can be any bus master like CPU or DMA).

PWM mode

The LPTIM can be configured to autonomously change, at each update event, the output waveform pulse width and/or the duty cycle without any CPU intervention. To enable this autonomous mode, the corresponding UEDE bit must be set in the LPTIM_DIER register.

At each update event, the APB clock is requested by the peripheral and a DMA request is generated. DMA direction must be configured as memory-to-peripheral which enables updating LPTIM registers, at each DMA request, with values stored in SRAM.

The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA. Thus, to enable automatic hardware clearing of UE flag, the application must configure the LPTIM_ARR register to be the last one to be written (at the end of list). For instance, if LPTIM_CCR1 and LPTIM_RCR registers need to be updated in Stop mode by DMA, the update sequence must be: LPTIM_CCR1, LPTIM_RCR then LPTIM_ARR.

The UE flag can also be cleared over its corresponding clear bit UECF in the LPTIM_ICR register, this can be done by configuring the DMA to write the LPTIM_ICR register at the end of register update.

58.4.21 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

58.4.22 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

58.5 LPTIM low-power modes

58.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 605. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 58.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 58.3: LPTIM implementation*.
3. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 58.3: LPTIM implementation*.

58.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 167* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

58.7.1 LPTIM5 interrupt and status register (LPTIM5_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

58.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 4)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

58.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 4)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OF	CC1 OF	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	Res.	EXT TRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

58.7.4 LPTIM5 interrupt clear register (LPTIM5_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK CF	UECF	DOWN CF	UPCF	ARRO KCF	CMP1 OKCF	EXTTR IGCF	ARRM CF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 4)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 4)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OCF	CC1OCF	Res.	Res.	CC2CF	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.7 LPTIM5 interrupt enable register (LPTIM5_DIER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 4)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 4)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers
Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 58.3.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: Software trigger (counting start is initiated by software)
01: Rising edge is the active edge
10: Falling edge is the active edge
11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 58.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger
01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition
01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 58.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

58.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

58.7.12 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

58.7.13 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

58.7.14 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value
When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

58.7.15 LPTIM1 option register (LPTIM1_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_0
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OR_0**: Option register bit 0
0: LPTIM1 channel1 output is connected to I/O or other function
1: LPTIM1 channel1 output is connected to TAMP

58.7.16 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection
The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection
The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection
The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection
The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

58.7.17 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

58.7.18 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 58.3.

58.7.19 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 58.3: LPTIM implementation*.

58.7.20 LPTIM peripheral hardware configuration register 2 (LPTIM_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0001 0402 (for LPTIM1/2/3/4)

Reset value: 0x0000 0400 (for LPTIM5)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3 [16]
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								Res.	Res.	Res.	Res.	CFG1[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CFG3[16]**: peripheral hardware configuration 3

This is read only bitfield and is read as CFG3[16] = autonomous mode.

0: disabled

1: enabled

Bits 15:8 **CFG2[7:0]**: peripheral hardware configuration 2

This is read only bitfield and is read as CFG2[7:0] = input capture multiplexer size.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **CFG1[3:0]**: peripheral hardware configuration 1

This is read only bitfield and is read as CFG1[3:0] = channel number.

58.7.21 LPTIM peripheral hardware configuration register 1 (LPTIM_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0101 0804 (for LPTIM1)

Reset value: 0x0001 0804 (for LPTIM2)

Reset value: 0x0000 0804 (for LPTIM3/4/5)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG4[7:0]								Res.	Res.	Res.	Res.	CFG3[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								CFG1[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **CFG4[7:0]**: peripheral hardware configuration 4
This is read only bitfield and is read as CFG4[7:0] = option register.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CFG3[3:0]**: peripheral hardware configuration 3
This is read only bitfield and is read as CFG3[3:0] = Encoder mode (1: enabled, 0: disabled).

Bits 15:8 **CFG2[7:0]**: peripheral hardware configuration 2
This is read only bitfield and is read as CFG2[7:0] = trigger number.

Bits 7:0 **CFG1[7:0]**: peripheral hardware configuration 1
This is read only bitfield and is read as CFG1[7:0] = input multiplexer size.

58.7.22 LPTIM peripheral version identification register (LPTIM_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[7:4]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[7:4]**: Major revision
This is read only bitfield and is read as MAJREV[7:4] = 2.

Bits 3:0 **MINREV[3:0]**: Minor revision
This is a read only bitfield and is read as MINREV[3:0] = 3.

58.7.23 LPTIM peripheral type identification register (LPTIM_PIDR)

Address offset: 0x3F8

Reset value: 0x0012 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **P_ID[31:0]**: Peripheral type identifier

This a read only register and is always read as P_ID[31:0] = 0x00120011.

58.7.24 LPTIM registers map size identification register (LPTIM_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **S_ID[31:0]**: Registers map size identifier

This a read only register and is always read as S_ID[31:0] = 0xA3C5DD01.

This register identifies the size of the memory region allocated to the LPTIM registers. The size of this memory region is 1 KB.

58.7.25 LPTIM register map

The following table summarizes the LPTIM registers.

Table 606. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM5_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0

Table 606. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 1 to 4) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 4) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM5_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0																0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 4) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 4) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM5_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIMx_DIER (x = 1 to 4) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 4) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	WAVPOL(3)	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]		Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL	
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 606. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	LPTIM1_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_0
	Reset value																																0
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1(4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	LPTIMx_ HWCFGR2 (x = 1 to 4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[16]	CFG2[7:0]								Res.	Res.	Res.	Res.	CFG1 [3:0]			
	Reset value																1	0	1	0	0	0	0	0	0					0	0	1	0
0x3EC	LPTIM5_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[16]	CFG2[7:0]								Res.	Res.	Res.	Res.	CFG1 [3:0]			
	Reset value																0	0	1	0	0	0	0	0	0					0	0	1	0
0x3F0	LPTIM1_ HWCFGR1	CFG4[7:0]								Res.	Res.	Res.	Res.	CFG3 [3:0]				CFG2[7:0]								CFG1[7:0]							
	Reset value	0	0	0	0	0	0	0	1					0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0x3F0	LPTIM2_ HWCFGR1	CFG4[7:0]								Res.	Res.	Res.	Res.	CFG3 [3:0]				CFG2[7:0]								CFG1[7:0]							
	Reset value	0	0	0	0	0	0	0	0					0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0x3F0	LPTIMx_ HWCFGR1 (x = 3 to 5)	CFG4[7:0]								Res.	Res.	Res.	Res.	CFG3 [3:0]				CFG2[7:0]								CFG1[7:0]							
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0x3F4	LPTIM_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [7:4]				MINREV [3:0]			
	Reset value																									0	0	1	0	0	0	1	1
0x3F8	LPTIM_PIDR	P_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	LPTIM_SIDR	S_ID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.

2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 58.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 58.3: LPTIM implementation*.

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

59 System timer generator (STGEN)

59.1 STGEN introduction

The STGEN generates a time-count value that provides a consistent view of time for multiple processors and other blocks in a device.

59.2 STGEN main features

- 64-bit wide to avoid roll-over issues
- Starts from 0 or a programmable value
- Control APB interface that enables the timer to be saved and restored across power-down events
- Read-only APB interface that enables the timer value to be read by non-secure software and debug tools
- hltdbg input to stop the timer value incrementing during full-system debug

59.3 STGEN functional description

The Cortex®-A35 generic timer schedules events and triggers interrupts based on an incrementing system counter (STGEN) value.

The generic timer provides:

- generation of timer events as interrupt outputs
- generation of event streams
- support for virtualization extensions

The Cortex®-A35 generic timer is compliant with the following Arm architecture reference manual system counter requirements:

- width: at least 56-bit wide
- frequency: increments at a fixed frequency

  The STGEN clock is simply derived from the external oscillator (HSE) in most circumstances, with the following exceptions:

  – During boot phase, the STGEN is clocked by HSI (64 MHz) until HSE is set up.

  – During low-power modes (Stop with HSE off and Standby), the STGEN is stopped by software. It is enabled after returning from these low-power modes, with time adjustment by secure software according to the RTC.
- accuracy: the Arm architecture reference manual does not specify a required accuracy, but recommends that the counter does not gain or lose more than ten seconds in a 24-hour period. This requirement is supported by software with occasional STGEN update (by increment) according to the RTC.

  The use of a lower-frequency mode must not affect the implemented accuracy.
- start-up: the counter starts from 0 at POR.

The STGEN provides a uniform view of system time.

Note: Arm recommends that the system counter is in an always-on power domain, which is not supported in the current implementation. The STGEN must then be saved and restored before Standby mode entry, and restored at Standby exit by secure software.

During low-power modes, the counter may increment by larger steps using a lower frequency. In this case, the counter must support transitions between high-frequency high-precision operation, and lower-frequency lower-precision operation, without any impact on the required counter accuracy.

The software can access the STGENC_CNTFIDO register to read the clock frequency of the system counter, and to modify the value of this register.

59.3.1 STGEN block diagram

The figure below shows the STGEN overview, with:

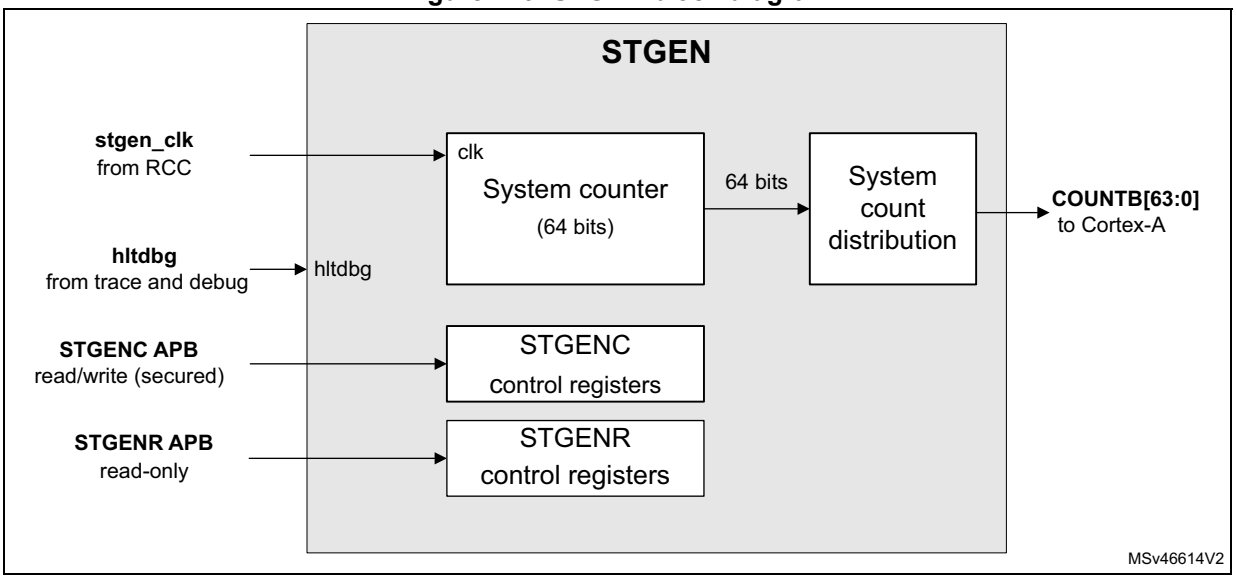
- a 64-bit system counter
- control registers with two APB ports, corresponding to STGENC and STGENR registers
- system count distribution logic to format the system counter value to Cortex®-A35 with gray encoding and frequency interpolation.

Note: The two APB ports are used because of TrustZone security isolation as follows:

- STGENC is read/write only by Cortex®-A35 secure and privilege.

- STGENR is read only by all CID, secure or non-secure, privileged or unprivileged.

Figure 719. STGEN block diagram



Note: hltdbg is a combinatorial AND of DBGACK outputs from Cortex®-A35 processors, so that the STGEN is halted when both Cortex®-A35 processors are in debug.

59.4 STGEN registers

STGEN control registers are organized in two sets accessible by two APB interfaces:

- The first set accessible as STGEN**C** APB is read/write and made TrustZone secured.
- The second set accessible by STGEN**R** APB is read-only and not concerned by TrustZone security.

59.4.1 STGENC control register (STGENC_CNTCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HLTDBG	EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HLTDBG:** Halt on debug

0: Do not halt on debug; hltdbg signal into the counter has no effect.

1: Halt on debug; when hltdbg is driven high, the count value is held static.

Bit 0 **EN:** Enable

0: Counter disabled and not incrementing

1: Counter enabled and incrementing

59.4.2 STGENC status register (STGENC_CNTSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBGH:** Debug halted

0: Normal operation

1: Counter halted due to system-wide debug state

Bit 0 Reserved, must be kept at reset value.

59.4.3 STGENC value lower register (STGENC_CNTCVL)

Address offset: 0x008

Reset value: 0x0000 0000

Note: The control interface must clear the STGENC_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVL_L_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVL_L_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVL_L_32[31:0]**: Current value of the timestamp counter, lower 32 bits

To change the current timestamp value, write the lower 32 bits of the new value to this register before writing the upper 32 bits to STGENC_CNTCVU. The timestamp value is not changed until the STGENC_CNTCVU register is written to.

59.4.4 STGENC value upper register (STGENC_CNTCVU)

Address offset: 0x00C

Reset value: 0x0000 0000

Note: The control interface must clear the STGENC_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_U_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_U_32[31:0]**: Current value of the timestamp counter, upper 32 bits

To change the current timestamp value, write the lower 32 bits of the new value to STGENC_CNTCVL before writing the upper 32 bits to this register. The 64-bit timestamp value is updated with the value from both writes when this register is written to.

59.4.5 STGENC base frequency register (STGENC_CNTFID0)

Address offset: 0x020

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]:** Frequency in number of ticks per second

It can be specified up to 4 GHz. This register must be programmed to match the clock frequency of the clock input to TSGEN (tsgen_clk), in ticks per second. For example, for a 50 MHz clock, the register value has to be 0x02FAF080.

59.4.6 STGENC peripheral ID4 register (STGENC_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]:** Always 0b0000

Indicates that the device only occupies 4 Kbytes of memory.

Bits 3:0 **DES_2[3:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

0b0100 JEDEC continuation code

59.4.7 STGENC peripheral ID5 register (STGENC_PIDR5)

Address offset: 0xFD4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR5[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR5[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR5[31:0]:** Peripheral ID5

59.4.8 STGENC peripheral ID6 register (STGENC_PIDR6)

Address offset: 0xFD8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR6[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR6[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR6[31:0]:** Peripheral ID6

59.4.9 STGENC peripheral ID7 register (STGENC_PIDR7)

Address offset: 0xFDC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR7[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR7[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR7[31:0]:** Peripheral ID7

59.4.10 STGENC peripheral ID0 register (STGENC_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]:** Bits[7:0] of the 12-bit part number of the component
The designer of the component assigns this part number.

59.4.11 STGENC peripheral ID1 register (STGENC_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]:** Part of designer identification
Together PIDR1.DES_0, PIDR2.DES_1, and PIDR4.DES_2 identify the designer of the component.

Bits 3:0 **PART_1[3:0]:** Bits[11:8] of the 12-bit part number of the component
The designer of the component assigns this part number.

59.4.12 STGENC peripheral ID2 register (STGENC_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** device revision number

Bit 3 **JEDEC:** always 1
Indicates that the JEDEC-assigned designer ID is used.

Bits 2:0 **DES_1[2:0]:** part of designer identification
Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

59.4.13 STGENC peripheral ID3 register (STGENC_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Customer version
0b0000 indicates there are no errata fixes to this component.

Bits 3:0 **CMOD[3:0]:** Customer modified
Indicates whether the customer has modified the behavior of the component. In most cases, this bitfield is 0b0000 (no modification done). Customers change this value when they make authorized modifications to this component.

59.4.14 STGENC component ID0 register (STGENC_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]:** Preamble 0
Contains bits[7:0] of the component identification code.

59.4.15 STGENC component ID1 register (STGENC_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Class of the component

For example, the component can be a ROM table or a generic CoreSight™ SoC-400 component. Contains bits[15:12] of the component identification code.
0b1111 indicates the component is a CoreSight™ SoC-400 component.

Bits 3:0 **PRMBL_1[3:0]:** preamble 1

Contains bits[11:8] of the component identification code.

59.4.16 STGENC component ID2 register (STGENC_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]:** Preamble 2

Contains bits[23:16] of the component identification code.

59.4.17 STGENC component ID3 register (STGENC_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]:** Preamble 3
Contains bits[31:24] of the component identification code.

59.4.18 STGENR value lower register (STGENR_CNTCVL)

Address offset: 0x000

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVL_L_32[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVL_L_32[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNTCVL_L_32[31:0]**: Current value of the timestamp counter (the lower 32 bits)

59.4.19 STGENR value upper register (STGENR_CNTCVU)

Address offset: 0x004

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_U_32[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNTCVU_U_32[31:0]**: Current value of the timestamp counter (the upper 32 bits)

59.4.20 STGENR peripheral ID4 register (STGENR_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]:** Always 0b0000

Indicates that the device only occupies 4 Kbytes of memory.

Bits 3:0 **DES_2[3:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

59.4.21 STGENR peripheral ID5 register (STGENR_PIDR5)

Address offset: 0xFD4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR5[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR5[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR5[31:0]:** Peripheral ID5

59.4.22 STGENR peripheral ID6 register (STGENR_PIDR6)

Address offset: 0xFD8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR6[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR6[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR6[31:0]:** Peripheral ID6

59.4.23 STGENR peripheral ID7 register (STGENR_PIDR7)

Address offset: 0xFDC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR7[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR7[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR7[31:0]:** Peripheral ID7

59.4.24 STGENR peripheral ID0 register (STGENR_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]:** Bits[7:0] of the 12-bit part number of the component

59.4.25 STGENR peripheral ID1 register (STGENR_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

Bits 3:0 **PART_1[3:0]:** Bits[11:8] of the 12-bit part number of the component

59.4.26 STGENR peripheral ID2 register (STGENR_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Device revision number

Bit 3 **JEDEC:** Always 1

Indicates that the JEDEC-assigned designer ID is used.

Bits 2:0 **DES_1[2:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2, identify the designer of the component.

59.4.27 STGENR peripheral ID3 register (STGENR_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Errata fix identification

0b0000 indicates there are no errata fixes to this component.

Bits 3:0 **CMOD[3:0]:** Customer modified

Indicates whether the customer has modified the behavior of the component. In most cases, this bitfield is 0b0000 (no modification done). Customers change this value when they make authorized modifications to this component.

59.4.28 STGENR component ID0 register (STGENR_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]:** Preamble 0
Contains bits[7:0] of the component identification code.

59.4.29 STGENR component ID1 register (STGENR_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Class of the component
Contains bits[15:12] of the component identification code.

Bits 3:0 **PRMBL_1[3:0]:** Preamble 1
Contains bits[11:8] of the component identification code.

59.4.30 STGENR component ID2 register (STGENR_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]:** Preamble 2
Contains bits[23:16] of the component identification code.

59.4.31 STGENR component ID3 register (STGENR_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]:** Preamble 3
Contains bits[31:24] of the component identification code.

59.4.32 STGEN register map

Table 607. STGENC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	STGENC_CNTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HLTDBG	EN
	Reset value																															0	0
0x004	STGENC_CNTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
	Reset value																															0	
0x008	STGENC_CNTCVL	CNTCVL_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	STGENC_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0100-0x01C	Reserved	Reserved																															
0x0020	STGENC_CNTFID0	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x0FCC	Reserved	Reserved																															
0xFD0	STGENC_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 + 0x4*(x-5) (x=5 to 7)	STGENC_PIDRx	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0	STGENC_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	0	0	0	0	0	0	1

Table 607. STGENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	STGENC_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	STGENC_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	DES_1 [2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	STGENC_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STGENC_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	STGENC_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1 [3:0]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	STGENC_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	STGENC_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Table 608. STGENR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	STGENR_CNTCVL	CNTCVL_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	STGENR_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008-0xFCC	Reserved	Reserved																															
0xFD0	STGENR_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 + 0x4*(x-5) (x=5 to 7)	STGENR_PIDR5x	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0-0xFDC	Reserved	Reserved																															
0xFE0	STGENR_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	STGENR_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	STGENR_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	DES_1 [2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	STGENR_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STGENR_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	STGENR_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1 [3:0]			
	Reset value																									1	1	1	1	0	0	0	0

Table 608. STGENR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	STGENR_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
	Reset value																									0	1	0	1	0	0	0	0
0xFFC	STGENR_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

60 Watchdog overview

The STM32MP21xx device includes watchdogs which offer a combination of high safety, timing accuracy and flexibility. In the whole document:

- The Cortex-A35 is also called CPU1 or MPU.
- The Cortex-M33 is also called CPU2 or MCU.

60.1 Watchdog main features

- Two independent watchdogs (IWDG1/2) dedicated to CPU1
  The IWDG1 is dedicated to CPU1 secure environment (TrustZone securable).
- Two independent watchdogs (IWDG3/4) dedicated to CPU2
  The IWDG3 is dedicated to CPU2 secure environment (TrustZone securable).
- One window watchdog (WWDG1) dedicated to CPU2
- The watchdogs can be configured to generate an early wake-up interrupt.
- Independent watchdogs can generate a system reset or a notification in case of timeout or refresh outside the expected window.
- The watchdog behavior can be configured via option bytes (hardware auto-start mode and freeze in low-power modes), and debug option (freeze in debug mode).

60.2 Watchdog functional description

All watchdog peripherals (independent and window) are used to detect and resolve malfunctions due to software or hardware failures.

Figure 720 shows the interconnection of the independent watchdogs into the system.

The independent watchdogs (IWDG1 to IWDG4) are clocked by the low-speed clock (LSI), and thus stay active even if the main clock fails. In this way, they are best suited to applications which require the watchdog to run as a totally independent process outside the main application. The IWDGs are best suited to recover from unexpected software or hardware failures.

IWDGs may generate a system reset when a timeout occurs (rst_iwdg[4:1] signal) according to the application selection. This signal is also routed to the EXTI in order to wake up another CPU to handle the error.

Either IWDG1 when tdcid = 0x1 or IWDG3 when tdcid = 0x2, generates a system reset when the timeout occurs. The reset signal generates also a tamper event when the timeout occurs (rst_iwdg[3,1] signal). See *Section 20.3.5: Watchdog resets* for additional information.

Figure 720. Top view of independent watchdogs

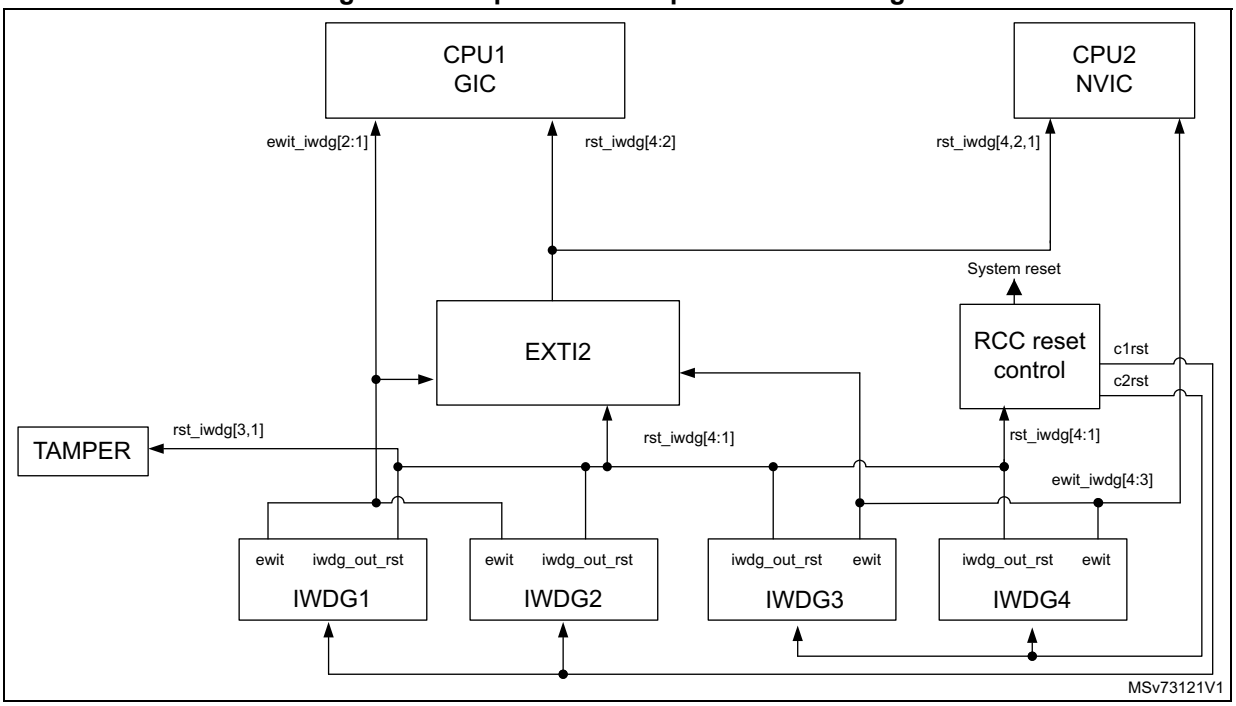
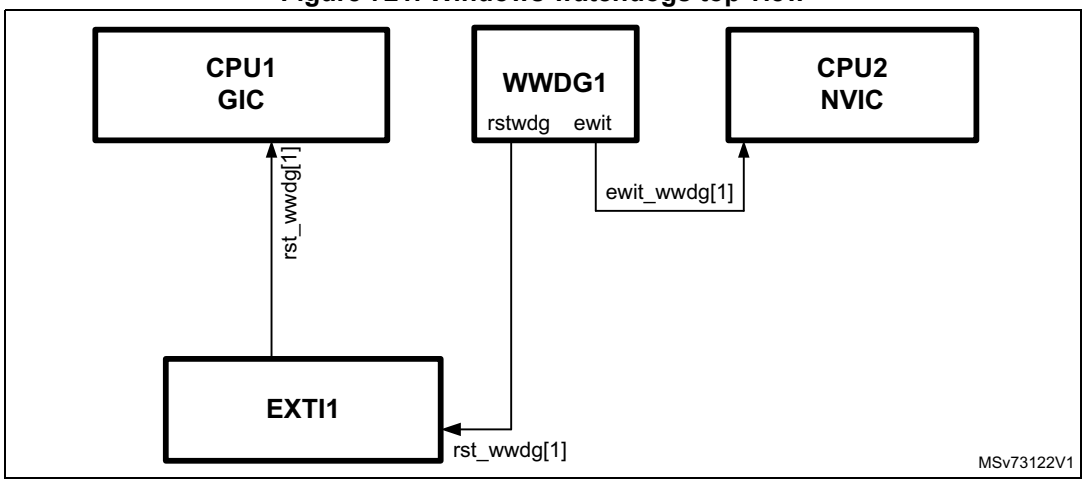


Figure 721 shows the interconnection of the window watchdogs into the system.

The WWDG1 is clocked with the APB3 clock, and provides a reset and an early interrupt (ewit_wwdg[1] signal) that is connected to the CPU2 (NVIC) interrupt controller.

The WWDG1 also generates a reset signal when a timeout occurs (rst_wwdg[1] signal). This signal is only routed to the EXTI in order to wake up the CPU1 core to handle emergency tasks if needed.

Figure 721. Windows watchdogs top view



61 Independent watchdog (IWDG)

61.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 61.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

61.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

61.3 IWDG implementation

Table 609. STM32MP21xx IWDG features (1)

IWDG modes/features	IWDG1	IWDG2	IWDG3	IWDG4
LSI used as IWDG kernel clock (iwdg_ker_ck)	X	X	X	X
Window function	X	X	X	X
Early wake-up interrupt generation	X	X	X	X
System reset generation (2)	X	X	X	X
Capability to work in system Stop	X	X	X	X
Capability to work in system Standby	X	X	X	X
Capability to generate an interrupt in system Stop	X	X	X	X
Capability to generate an interrupt in system Standby	-	-	-	-
Capability to be frozen when the CPU1 enters in debug mode	X	X	-	-
Capability to be frozen when the CPU2 enters in debug mode	-	-	X	X
Option bytes to control the activity in Stop mode	X	X	X	X
Option bytes to control the activity in Standby mode	X	X	X	X
Option bytes to control the hardware mode	X	X	X	X

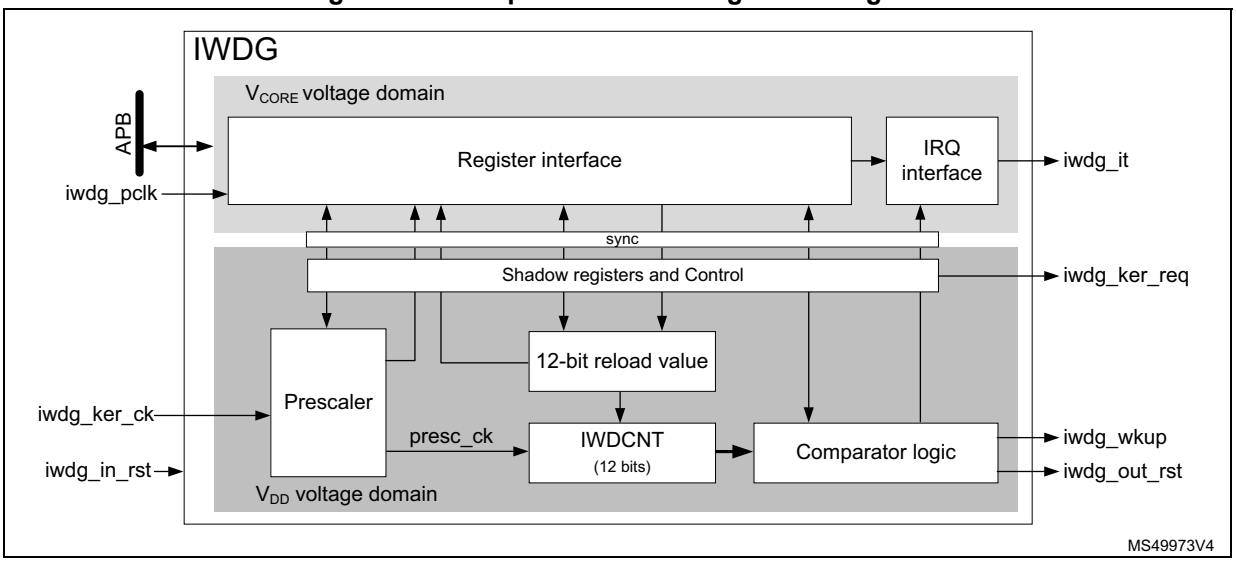
1. 'X' = supported, '-' = not supported.
2. Refer to the Watchdogs overview section for additional information.

61.4 IWDG functional description

61.4.1 IWDG block diagram

Figure 722 shows the functional blocks of the independent watchdog module.

Figure 722. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 61.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 610. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

61.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 611*.

Table 611. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

61.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 61.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

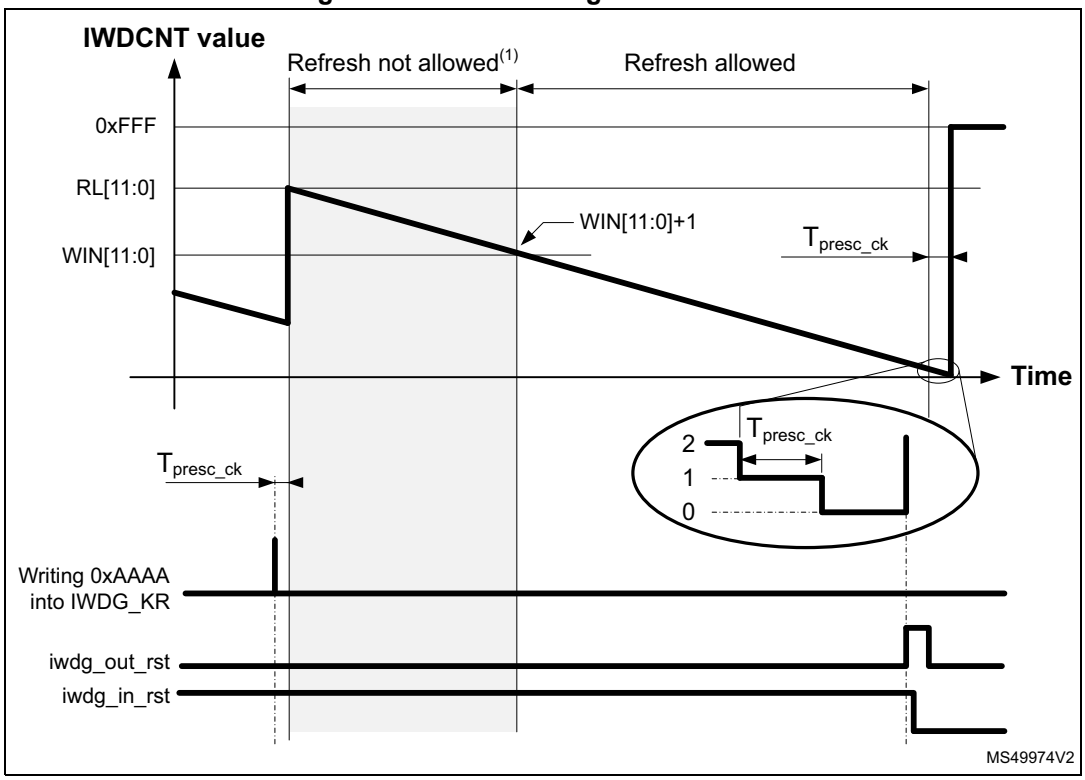
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 723*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 723. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

61.4.4 Window option

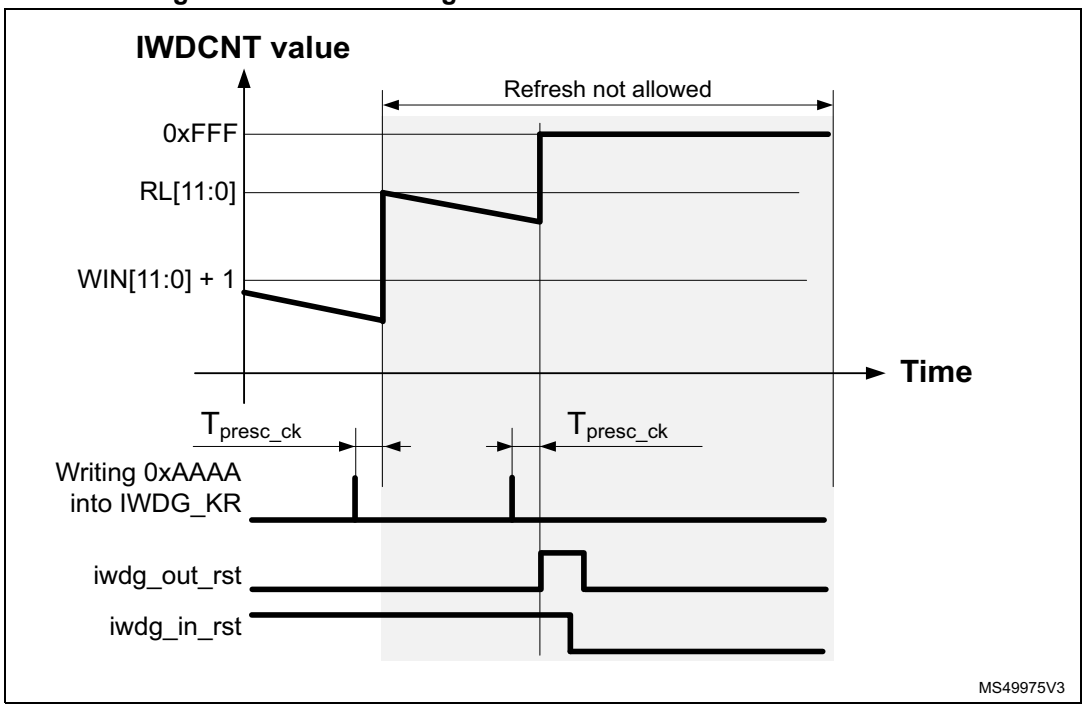
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 724*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 724. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

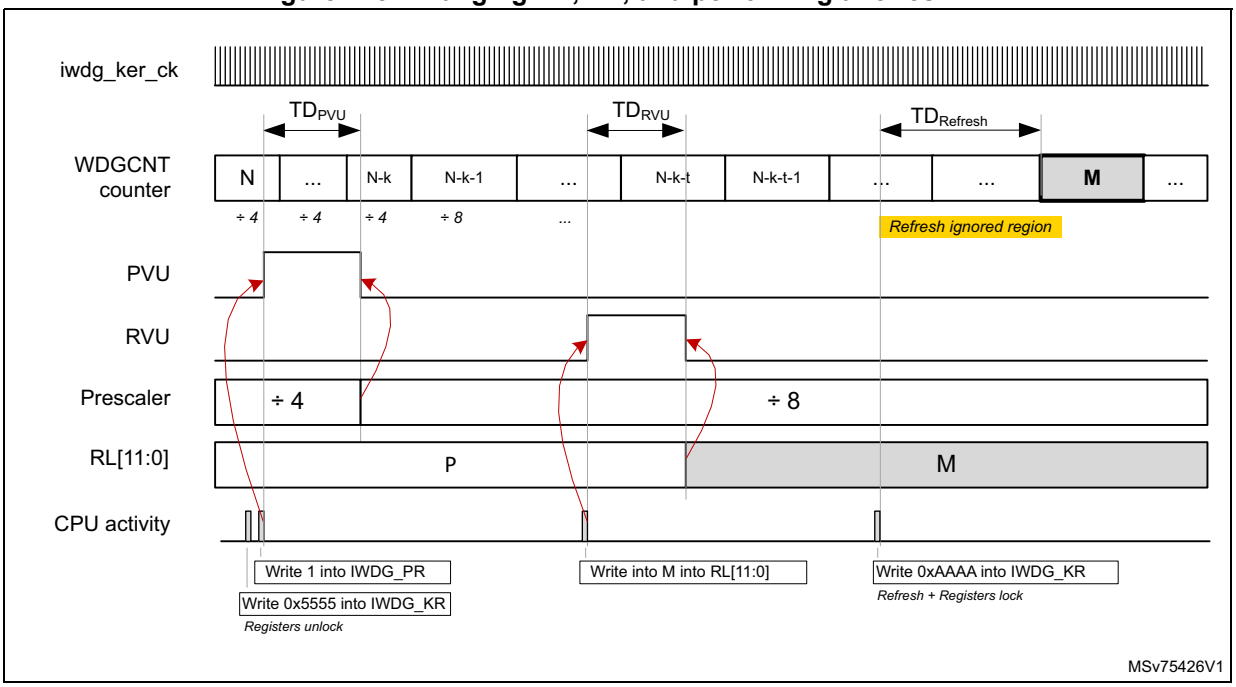
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 725. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 610: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

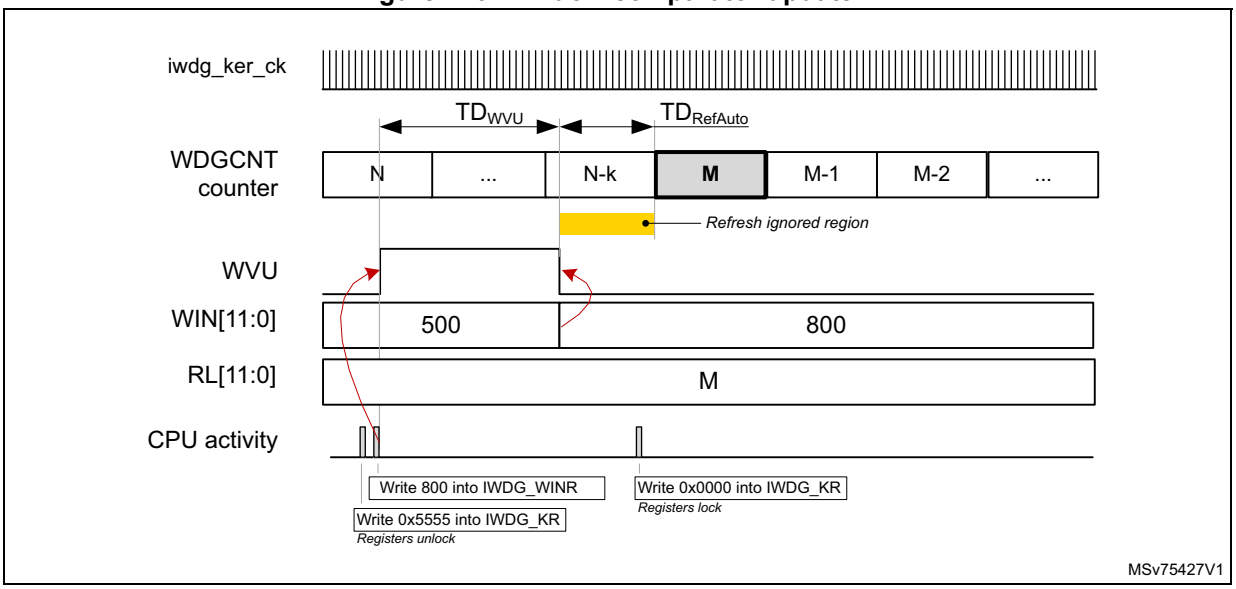
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 726 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 726. Window comparator update(1)



1. Refer to *Table 610: IWDG delays versus actions* for details on timing values.

61.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 61.3* for details on the capabilities of this product.

61.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

61.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 61.3* for details.

Table 612. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 61.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 61.3* for details. IWDG interrupts do not make the device to exit from Standby mode.
Shutdown	The IWDG is not working.

61.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

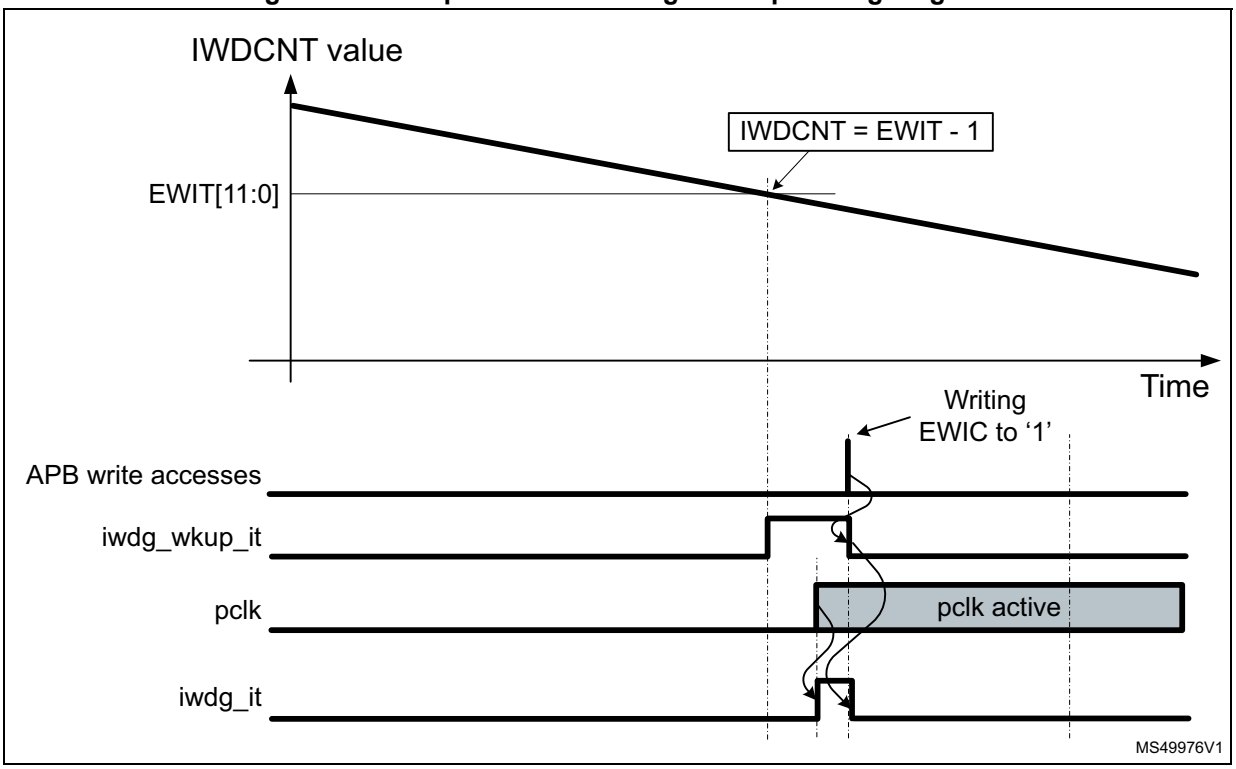
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG interrupt clear register (IWDG_ICR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 727. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

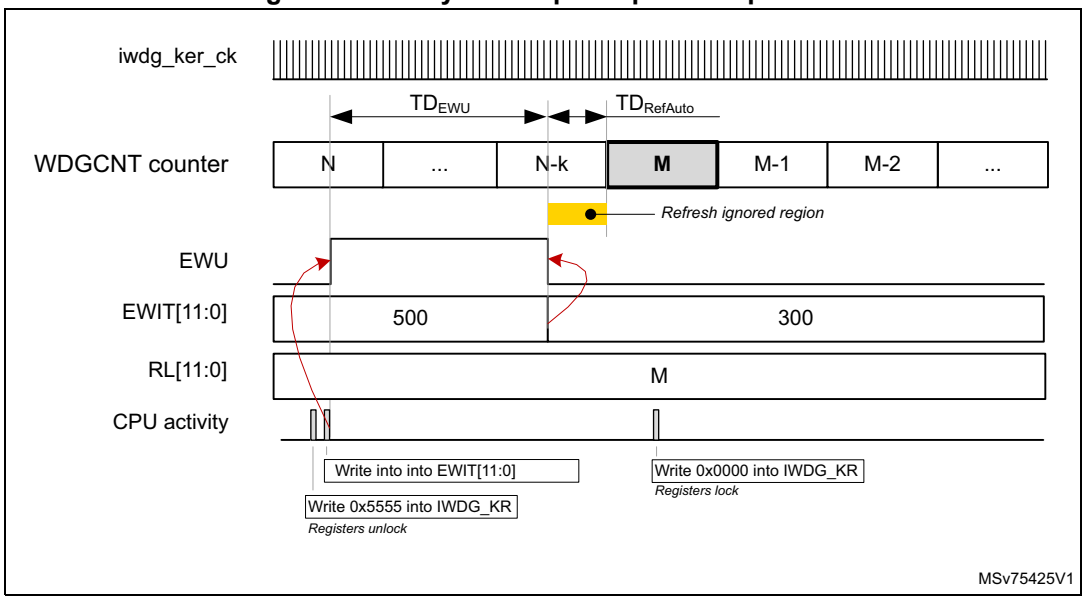
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 728 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 728. Early wake-up comparator update(1)



1. Refer to *Table 610: IWDG delays versus actions* for details on timing values.

Table 613 summarizes the IWDG interrupt request.

Table 613. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

61.7 IWDG registers

Refer to *Section 1.2 on page 167* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

61.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

61.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 61.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

61.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 61.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

61.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
r							r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_ICR register to ‘1’.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 610: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 610: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 610: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 610: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

61.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 61.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

61.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	Res.	Res.	Res.	EWIT[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 61.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

61.7.7 IWDG interrupt clear register (IWDG_ICR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 14:0 Reserved, must be kept at reset value.

61.7.8 IWDG hardware configuration register (IWDG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 00F1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR_DEFAULT[3:0]				WINDOW[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PR_DEFAULT[3:0]**: Prescaler default value
This bitfield gives the default value of the prescaler (see *PR[3:0]: Prescaler divider* of *IWDG prescaler register (IWDG_PR)*
0000: Prescaler (PR[3:0]) default value after reset is 0000'
0001: Prescaler (PR[3:0]) default value after reset is 0001'
0010: Prescaler (PR[3:0]) default value after reset is 0010'
0011: Prescaler (PR[3:0]) default value after reset is 0011'
0100: Prescaler (PR[3:0]) default value after reset is 0100'
0101: Prescaler (PR[3:0]) default value after reset is 0101'
0110: Prescaler (PR[3:0]) default value after reset is 0110'
0111: Prescaler (PR[3:0]) default value after reset is 0111'
1xxx: Prescaler (PR[3:0]) default value after reset is 1000

Bits 3:0 **WINDOW[3:0]**: Support of Window function
0000: Window option disabled
0001: Window option enabled
Others: Reserved

61.7.9 IWDG version register (IWDG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: IWDG major revision

Bits 3:0 **MINREV[3:0]**: IWDG minor revision

61.7.10 IWDG identification register (IWDG_IDR)

Address offset: 0x3F8

Reset value: 0x0012 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IWDG identifier

61.7.11 IWDG size identification register (IWDG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: IWDG size identifier

The address decoding range is 1 Kbyte.

61.7.12 IWDG register map

Table 614. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													1	1	1	1
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																	0							x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	Res.	Res.	Res.	EWIT[11:0]											
	Reset value																	0				0	0	0	0	0	0	0	0	0	0	0	0
0x18	IWDG_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0															
0x3F0	**IWDG_HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR_DEFAULT [3:0]				WINDOW [3:0]			
	Reset value																									1	1	1	1	0	0	0	1
0x3F4	**IWDG_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	0	0
0x3F8	**IWDG_IDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	**IWDG_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

62 System window watchdog (WWDG)

62.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

62.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 730*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

62.3 WWDG implementation

Table 615. STM32MP21xx WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters Debug mode(3)	X
Option bytes to control the hardware mode(4)	-

1. “X” = supported, “-” = not supported
2. Refer to the RCC section for additional information.
3. Controlled via DBG_WWDG_STOP in DBG blockbit of DBGMCU_APB3FZ1 register.

4. Controlled via the option byte WWDG_SW. When this byte is set in HW mode the WWDG runs as soon as the CPU is in Run or Sleep mode.

62.4 WWDG functional description

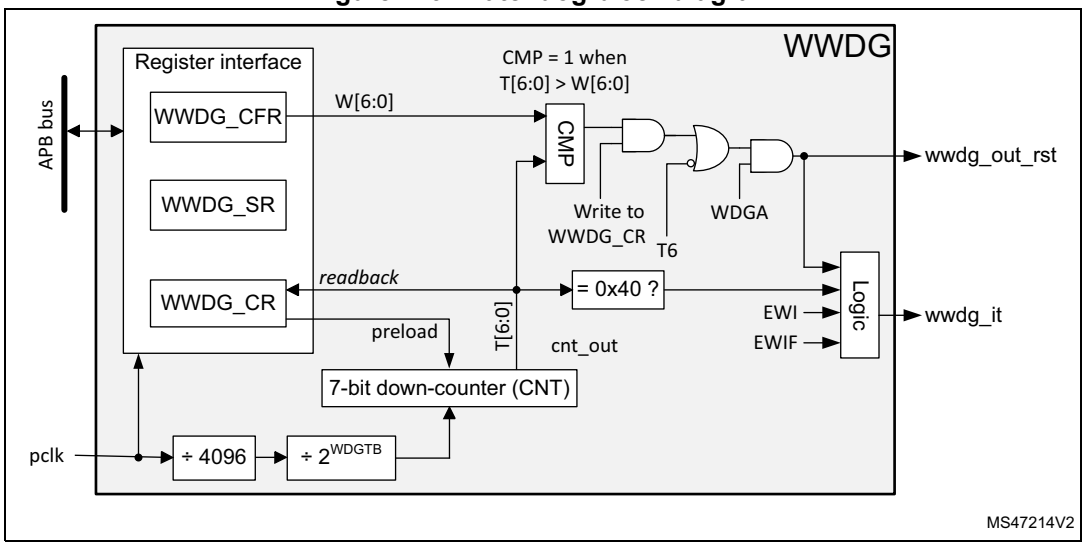
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 729* and to *Section 62.4.2: WWDG internal signals* for the WWDG block diagram.

62.4.1 WWDG block diagram

Figure 729. Watchdog block diagram



62.4.2 WWDG internal signals

Table 616 gives the list of WWDG internal signals.

Table 616. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

62.4.3 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

62.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 730*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 730* describes the window watchdog process.

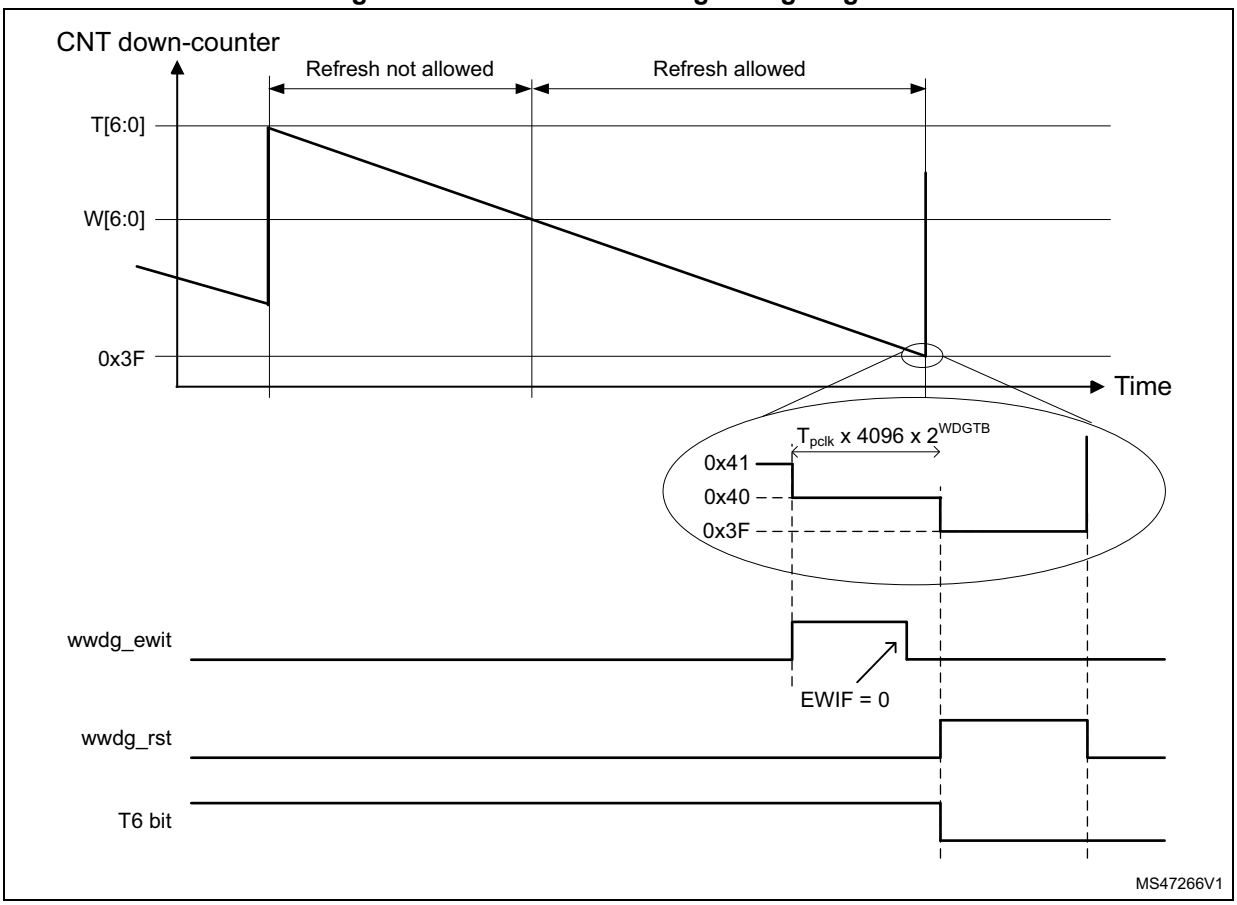
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

62.4.5 How to program the watchdog timeout

Use the formula in *Figure 730* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 730. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0]+1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63+1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

62.4.6 Debug mode

When the CPU enters debug mode, WWDG counter either continues to work normally or stops, depending on debug settings. For more details refer to *Section 62.3: WWDG implementation* and to *Section 80: Debug support (DBG)*.

62.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

62.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

62.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

62.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:
000: CK counter clock (PCLK div 4096) div 1
001: CK counter clock (PCLK div 4096) div 2
010: CK counter clock (PCLK div 4096) div 4
011: CK counter clock (PCLK div 4096) div 8
100: CK counter clock (PCLK div 4096) div 16
101: CK counter clock (PCLK div 4096) div 32
110: CK counter clock (PCLK div 4096) div 64
111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

62.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

62.6.4 WWDG hardware configuration register (WWDG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREDIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PREDIV[15:0]**: The watchdog clock is prescaled by 4096 (PREDIV[15:0]).

62.6.5 WWDG version register (WWDG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: WWDG major revision

Bits 3:0 **MINREV[3:0]**: WWDG minor revision

62.6.6 WWDG identification register (WWDG_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0051

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: WWDG peripheral identifier

62.6.7 WWDG size identification register (WWDG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: The address decoding range is 1 Kbyte

62.6.8 WWDG register map

The following table gives the WWDG register map and reset values.

Table 617. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0
0x00C-0x3EC	Reserved	Reserved																															
0x3F0	WWDG_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV[15:0]															
	Reset value																	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x3F4	WWDG_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	0	0	0	0	1

Table 617. WWDG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F8	WWDG_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
0x3FC	WWDG_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

63 Real-time clock (RTC)

63.1 RTC introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

63.2 RTC main features

The RTC supports the following features (see *Figure 731: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection
- Alarm A, alarm B, wake-up Timer and timestamp individual compartment-based isolation

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

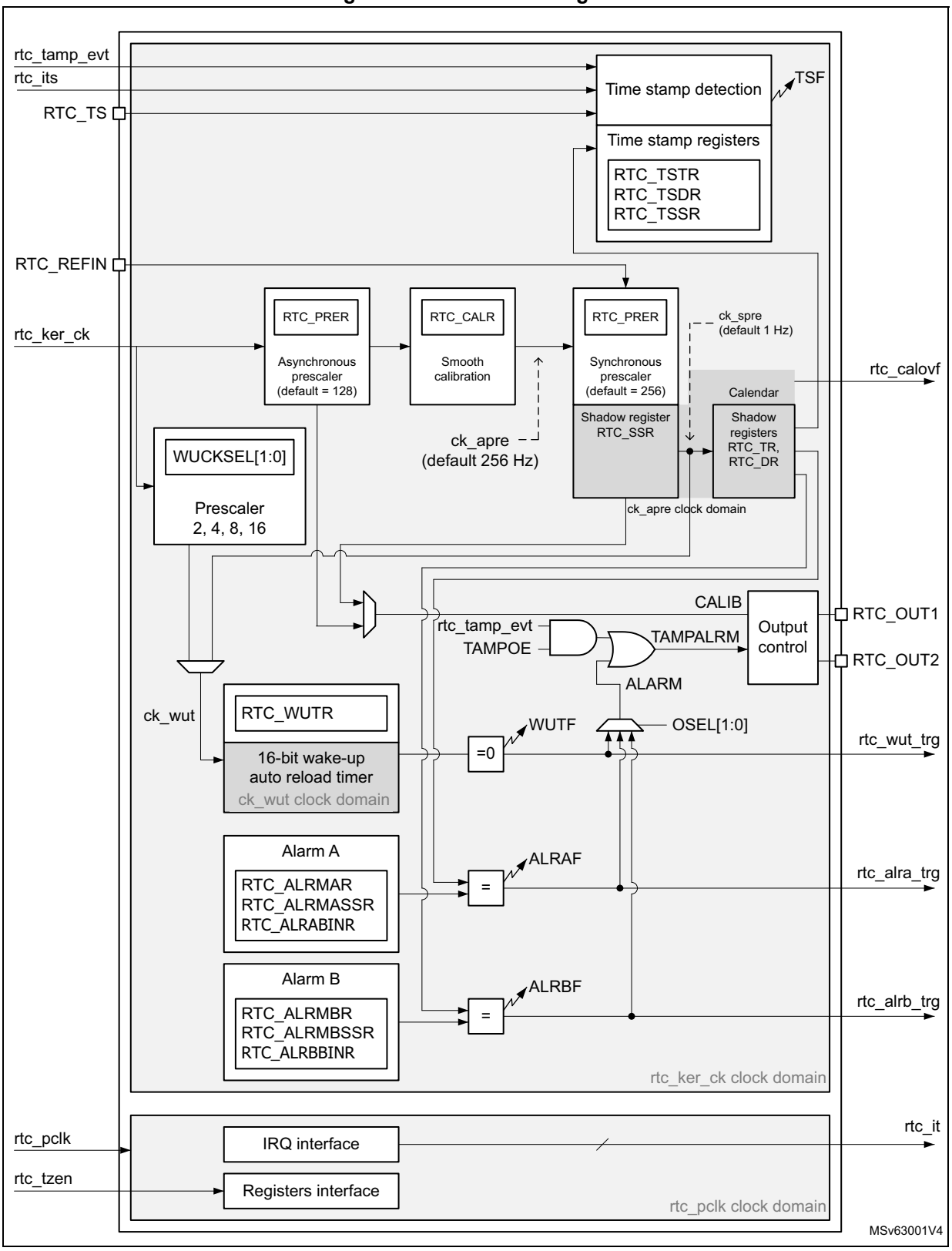
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

63.3 RTC functional description

63.3.1 RTC block diagram

Figure 731. RTC block diagram



63.3.2 RTC pins and internal signals

Table 618. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 619. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_dden	Input	Debug domain enable
rtc_ddcid[2:0]	Input	Debug domain CID
rtc_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 63.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 63.4: RTC low-power modes* for more details.

Table 620. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_tzen	Tied to 1
rtc_dden	From debug module
rtc_ddcid[2:0]	From debug module
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

63.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_LSO, RTC_TS and TAMP_OUT1 are mapped on the same pin (PC13). RTC_LSCO, the RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 621*.

Table 621. RTC pin PC13 configuration(1)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E=TAMP1AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=0	TSE (RTC_TS input enable)	LSCOEN[1:0]
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care
	00	1							
	01 or 10 or 11	1							

Table 621. RTC pin PC13 configuration(1) (continued)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E=TAMP1AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=0	TSE (RTC_TS input enable)	LSCOEN[1:0]
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT1 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
RTC_TS input floating		00	0	0	Don't care	Don't care	Don't care	0	1	Don't care
		00	0	1	1			0		
		Don't care	Don't care	0				0		
LSCO LSE clock output		00	0	0	Don't care	Don't care	Don't care	0	0	01
		00	0	1	1			0		
		Don't care	Don't care	0				0		

Table 621. RTC pin PC13 configuration(1) (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E=TAMP1AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=0	TSE (RTC_TS input enable)	LSCOEN[1:0]
Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	00 or 10 or 11
	00	0	1	1			0		
	Don't care	Don't care	0				0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

RTC_OUT2, TAMP_IN1, TAMP_OUT2 and RTC_LSCO are mapped on the same pin (PI8). PI8 configuration is controlled by the RTC, whatever the PI8 GPIO configuration. The RTC or TAMP functions mapped on PI8 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in *Table 622*.

Table 622. PI8 configuration

PI8 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN &OUT2_RMP	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E & IN1_RMP	LSCOEN[1:0]
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	1	0	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
TAMPALRM output Open-Drain	No pull	01 or 10 or 11	0	Don't care	1	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	1	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	1	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT2 output Push-Pull		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	0	Don't care	Don't care	0	1	Don't care
LSCO LSE clock output		00	0	0	0	Don't care	Don't care	0	0	10
Standard GPIO		00	0	0	0	Don't care	Don't care	0	0	00 or 01 or 11

In addition, it is possible to output RTC_OUT2 on PI8 pin thanks to OUT2EN bit. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 623*.

Table 623. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PI8
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

63.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) and the RTC Resource x CID configuration register (RTC_RxCIDCFGR, x=0 to 5) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_RxCIDCFGR(x = 0 to 5), RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

Caution: The SEC bit is forced by hardware to 0 when at least one CFEN is set in the RTC_RxCIDCFGR (x = 0 to 5) registers.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the

RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.

  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection, excepting RTC_WPR: a notification is generated through a flag/interrupt in the IAC (illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

63.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) and the RTC Resource x CID configuration register (RTC_RxCIDCFGR, x=0 to 5) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_RxCIDCFGR (x = 0 to 5), RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

Caution: The PRIV bit is forced by hardware to 0 when at least one CFEN is set in the RTC_RxCIDCFGR (x = 0 to 5) registers.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection, excepting RTC_WPR: a notification is generated through a flag/interrupt in the IAC (illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

63.3.6 RTC compartment-based isolation

By default after a backup domain power-on reset, compartment-based filters are disabled and any compartment has access to the RTC registers except for the Resource number x CID configuration registers (RTC_RxCIDCFGR, x=0 to 5) which can be written by the Trusted domain application only (TDCID secure privileged). The CID protection configuration is not affected by a system reset.

Table 624. RTC resource number allocation

RTC resource number	RTC Feature
0	Alarm A + Sub-second underflow
1	Alarm B
2	Wake-up timer
3	Timestamp
4	Calibration
5	Initialization

- When CFEN bit is set in the RTC_R0CIDCFGR:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE ALRAMSK and ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only by the CID defined in the RTC_R0CIDCFGR register.
  - ALRASEC in the RTC_SECCFGR and ALRAPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R0CIDCFGR register. They can be read by all CID.
- When CFEN bit is set in the RTC_R1CIDCFGR:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBMSK and ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only by the CID defined in the RTC_R1CIDCFGR register.
  - ALRBSEC in the RTC_SECCFGR and ALRBPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R1CIDCFGR register. They can be read by all CID.
- When CFEN bit is set in the RTC_R2CIDCFGR:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only by the CID defined in the RTC_R2CIDCFGR register.
  - WUTSEC in the RTC_SECCFGR and WUTPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R2CIDCFGR register. They can be read by all CID.
- When CFEN bit is set in the RTC_R3CIDCFGR:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF

in RTC_MISR and RTC_SMISR can be read and written only by the CID defined in the RTC_R3CIDCFGR register.

  - TSSEC in the RTC_SECCFGR and TSPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R3CIDCFGR register. They can be read by all CID.
- When CFEN bit is set in the RTC_R4CIDCFGR:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALSEC in the RTC_SECCFGR and CALPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R4CIDCFGR register.
  - These registers and control bits can be read by all CID.
- When CFEN bit is set in the RTC_R5CIDCFGR:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR and INITPRIV in the RTC_PRIVCFGR can be written only by the CID defined in the RTC_R5CIDCFGR register.
  - These registers and control bits can be read by all CID.

A non-allowed CID access to a CID-filtered register is denied:

- There is no bus error generated.
- In case the register has a global protection, excepting RTC_WPR: a notification is generated through a flag/interrupt in the IAC (illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

Note: CID protection is bypassed when DDEN = 1 and CID = DDCID (resources still protected by security and privilege).

63.3.7 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 731: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 63.3.11: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

63.3.8 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 63.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

63.3.9 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

63.3.10 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

63.3.11 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 3316*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 63.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

63.3.12 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After a system reset, the RTC registers are protected against parasitic write access by the DBD3P bit in the power control peripheral (refer to the PWR power control section). DBD3P bit must be set to enable RTC registers write access.

After Backup domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration, and by the CID configured in RTC_R5CIDCFGR register.

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege, security or CID settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration, and by the CID configured in RTC_R4CIDCFGR register.

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with the incorrect privilege, security and CID settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   – If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   – If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

63.3.13 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and

then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 3315*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 63.3.15: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

63.3.14 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

63.3.15 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

63.3.16 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same

moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

63.3.17 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP

to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

63.3.18 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

63.3.19 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

63.3.20 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

63.4 RTC low-power modes

Table 625. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 626. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	Yes	Yes

63.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration.

The RTC provides up to 8 interrupt signals: up to 4 RTC_CIDx (x=1 to 4) nonsecure interrupts and up to 4 RTC_S_CIDx (x=1 to 4) secure interrupts. Each interrupt signal CIDx can be connected to corresponding CPU. See *Section 63.3.6: RTC compartment-based isolation* for details.

Each feature provides an interrupt to the RTC_CIDx vector only if its configuration is nonsecure and if its CID filter is disabled, or enabled and configured with CID=x.

Each feature provides an interrupt to the RTC_S_CIDx vector only if its configuration is secure and if its CID filter is disabled, or enabled and configured with CID=x.

Table 627. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_CIDx (x=1 to 4)	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register. CPU selection (CFEN and CID bits) is configured in RTC_R0CIDCFGR for Alarm A and SSR underflow, RTC_R1CIDCFGR for Alarm B, RTC_R2CIDCFGR for wake-up timer, RTC_R3CIDCFGR for Timestamp.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 628. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag[1]	Enable control bit[2]	Interrupt clear method	Exit from low-power modes
RTC_S_CIDx (x=1 to 4)	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CALRAF	Yes[3]
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CALRBF	Yes[3]
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CTSF	Yes[3]
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CWUTF	Yes[3]
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1) and [CFEN=0 or (CFEN=1 and CID=x)]	write 1 in CSSRUF	Yes[3]

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register. CPU selection (CFEN and CID bits) is configured in RTC_R3CIDCFGR for Alarm A and SSR underflow, RTC_R4CIDCFGR for Alarm B, RTC_R5CIDCFGR for Alarm wake-up timer, RTC_R6CIDCFGR for Timestamp.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

63.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

63.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 3315* and *Reading the calendar on page 3316*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be write-protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

63.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 3315* and *Reading the calendar on page 3316*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be write-protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

63.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

63.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).

01: Free running Binary mode (BCD mode disabled)

10: Free running BCD calendar and Binary modes

11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

63.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 3315*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be write-protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

63.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

63.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 3314.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.

1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM

1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved

1: Save RTC timestamp on tamper detection event

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled

1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.

00: Output disabled

01: Alarm A output enabled

10: Alarm B output enabled

11: Wake-up output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 63.3.19: Calibration clock output*.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable

0: Wake-up timer interrupt disabled

1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable

0: Alarm B interrupt disable

1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable

0: Wake-up timer disabled

1: Wake-up timer enabled

Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable

0: SSR underflow interrupt disabled

1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

0: RTC_TS input rising edge generates a timestamp event

1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.

11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

63.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against

nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 63.3.5: RTC privilege protection modes*).

Each bit of this register can be individually write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Note: The PRIV bit is forced by hardware to 0 when at least one CFEN is set in the RTC_RxCIDCFGR (x = 0 to 5) registers.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 63.3.5: RTC privilege protection modes for details on the read protection.

63.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 63.3.5: RTC privilege protection modes*).

Each bit of this register can be individually write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Note: The SEC bit is forced by hardware to 0 when at least one CFEN is set in the RTC_RxCIDCFGR (x = 0 to 5) registers

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection

0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 63.3.4: RTC secure protection modes for details on the read protection.

63.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

63.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be write-protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be write-protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.

Refer to *Section 63.3.17: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 63.3.17: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 63.3.17: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 63.3.17: RTC smooth digital calibration on page 3319*.

63.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3314*.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

63.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

63.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

63.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

63.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

63.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

63.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

63.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

63.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.
If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.
If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

63.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag

This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp nonsecure masked flag

This flag is set by hardware when a timestamp on the internal event occurs and timestamp nonsecure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

63.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

63.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

63.6.24 RTC option register (RTC_OR)

Address offset: 0x60

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSCOEN[1:0]		OUT2_RMP
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:1 **LSCOEN[1:0]**: LSCO LSE clock output enable

00: no effect

01: LSE is output on PC13

10: LSE is output on PI8

11: no effect

Note: Note: LSCO function is output on PC13 or PI8 only if no other RTC output is enabled on these I/Os.

Bit 0 **OUT2_RMP**: RTC_OUT2 mapping

0: RTC_OUT2 is mapped on PF6 or PF4 (refer to datasheet)

1: RTC_OUT2 is mapped on PI8

63.6.25 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

63.6.26 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 63.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 63.3.5: RTC privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 63.3.6: RTC compartment-based isolation*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

63.6.27 RTC Resource x CID configuration register (RTC_RxCIDCFGR)

This register is used to configure the compartment ID filtering policy for the RTC local resource number x. Refer to *Section 63.3.6: RTC compartment-based isolation* for RTC resources number allocation and registers access conditions.

This register can be written only by the trusted domain CID when the APB access is secure and privileged.

Address offset: 0x80 + 0x04 * x, (x = 0 to 5)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]**: Compartment identifier
When CFEN=1, this field is used by the Trusted domain application to define which compartment has access to the RTC resource number x.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN**: Compartment ID filter enable
0: Compartment-based filtering is disabled for resource number x.
1: Compartment-based filtering is enabled for resource number x.

63.6.28 RTC hardware configuration register 2 (RTC_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0310

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CID_WIDTH[3:0]				RIF[3:0]				Res.	Res.	Res.	Res.
				r	r	r	r	r	r	r	r				

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **CID_WIDTH[3:0]**: CID length
0x1: CID length is 1 bit
0x2: CID length is 2 bits
0x3: CID length is 3 bits
0x4: CID length is 4 bits

Bits 7:4 **RIF[3:0]**: CID compartment filtering
0x0: CID compartment filtering disable
0x1: CID compartment filtering enable

Bits 3:0 Reserved, must be kept at reset value.

63.6.29 RTC hardware configuration register (RTC_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x1103 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BINARY[3:0]				TRUST_ZONE[3:0]				OPTIONREG_OUT[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[3:0]				SMOOTH_CALIB[3:0]				WAKEUP[3:0]				ALARMB[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **BINARY[3:0]**: Binary mode
- 0x0: Binary mode not implemented
- 0x1: Binary mode implemented

Bits 27:24 **TRUST_ZONE[3:0]**: Trust zone
- 0x0: Trust zone not implemented
- 0x1: Trust zone implemented

Bits 23:16 **OPTIONREG_OUT[7:0]**:
- 0: Configuration register not implemented
- 1: Configuration register bit 0 implemented
- 2: Configuration register bits 1:0 implemented
- ...
- 32: Configuration register bits 31:0 implemented

Bits 15:12 **TIMESTAMP[3:0]**:
- 0x0: Timestamp not implemented
- 0x1: Timestamp implemented

Bits 11:8 **SMOOTH_CALIB[3:0]**:
- 0x0: Smooth calibration not implemented
- 0x1: Smooth calibration implemented

Bits 7:4 **WAKEUP[3:0]**:
- 0x0: Wake-up timer not implemented
- 0x1: Wake-up timer implemented

Bits 3:0 **ALARMB[3:0]**:
- 0x0: Alarm B not implemented
- 0x1: Alarm B implemented

63.6.30 RTC version register (RTC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0035

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

63.6.31 RTC identification register (RTC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0033

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier

63.6.32 RTC size identification register (RTC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier

63.6.33 RTC register map

Table 629. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 629. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																		0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x60	RTC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSCOEN[1:0]		OUT2_RMP
	Reset value																														0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 629. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80+ 0x04*x,(x=0 to 5)	RTC_ RxCIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0
0x3EC	RTC_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID_WIDTH [3:0]				RIF[3:0]				Res.	Res.	Res.	Res.
	Reset value																					0	0	1	1	0	0	0	1				
0x3F0	RTC_ HWCFGR1	BINARY[3:0]				TRUST_ ZONE[3:0]				OPTION_REG_OUT[7:0]								TIMESTAMP [3:0]				SMOOTH_ CALIB[3:0]				WAKEUP [3:0]				ALARMB [3:0]			
	Reset value	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0x3F4	RTC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	1	0	1	0	1
0x3F8	RTC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
0x3FC	RTC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

64 Tamper and backup registers (TAMP)

64.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 128 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 12 tamper pins and 16 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

64.2 TAMP main features

- A tamper detection can optionally erase the backup registers, backup SRAM, SRAM1, and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 128×32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 12 tamper pins for 7 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 5 meshes (each input associated to its own exclusive output) to 7 meshes (single output shared for up to 7 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 16 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:

    1 read/write secure area.

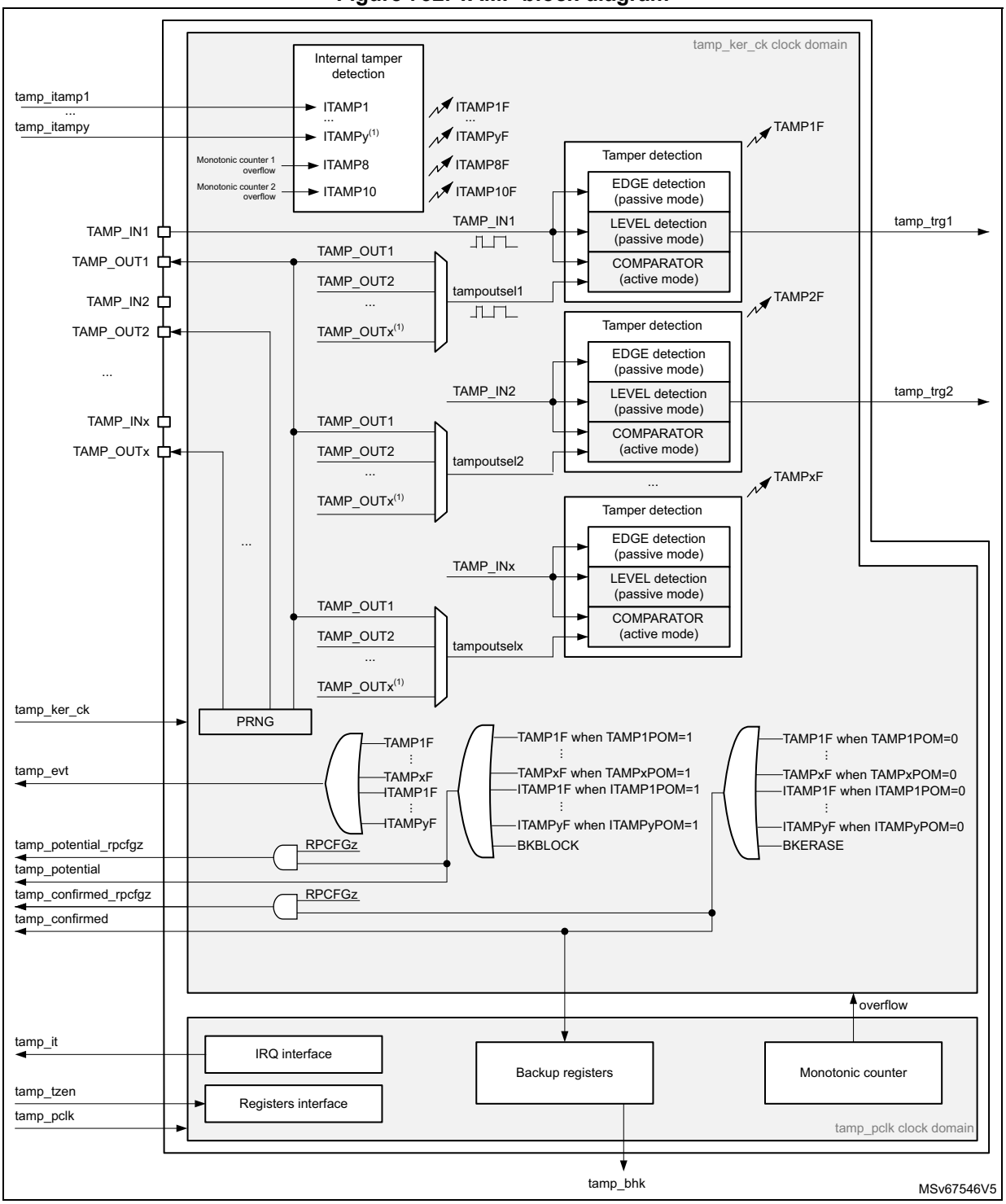
    1 write secure/read nonsecure area.

    1 read/write nonsecure area.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Tamper configuration and backup registers compartment-based isolation
- Two Monotonic counters.

64.3 TAMP functional description

64.3.1 TAMP block diagram

Figure 732. TAMP block diagram



1. The number of external and internal tampers depends on products.

64.3.2 TAMP pins and internal signals

Table 630. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 631. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_dden	Input	Debug domain enable
tamp_ddcid[2:0]	Input	Debug domain compartment identifier
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 631. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 64.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

1. Refer to *Table 632: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to for more details.

Table 632. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	Tied to 1
tamp_dden	From debug module
tamp_ddcid[2:0]	From debug module
tamp_evt	rtc_tamp_evt used to generate a timestamp event

Table 632. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_potential	The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – SRAM1 – BSEC (words 256 to 375) RHUK (root hardware unique key) in BSEC and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – SAES, CRYP1/2, HASH1, HASH2 peripherals, only when POTTAMPRESETMASK = 0 in SYSCFG (refer to *Section 23: System configuration controller (SYSCFG)*) – PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers – SRAM1 – ICACHE/DCACHE content – SAES, CRYP1/2, HASH1, HASH2 peripherals – OTFDEC 1/2 keys and CRC registers – PKA SRAM The device secrets access is blocked when erase is on-going. RHUK in BSEC (root hardware unique key) hardware bus to SAES is blocked.
tamp_potential_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_confirmed_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM(1) The device secrets access is blocked when erase is on-going.
tamp_itamp1	Backup domain voltage threshold monitoring(2)
tamp_itamp2	Temperature monitoring(2)
tamp_itamp3	LSE monitoring (LSECSS)(3)
tamp_itamp4	HSE monitoring (CSS + over frequency detection)(4)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG TAP access in secured-closed
tamp_itamp7	ADC2 (adc2_awd1) analog watchdog monitoring 1
tamp_itamp8(5)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (SAES or CRYP or PKA or RNG1/2)

Table 632. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_itamp10(5)	Monotonic counter 2 overflow
tamp_itamp11	IWDG3 reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC2 (adc2_awd2) analog watchdog monitoring 2
tamp_itamp13	ADC2 (adc2_awd3) analog watchdog monitoring 3
tamp_itamp14	RIFSC or BSEC or DBGMCU fault
tamp_itamp15	IWDG1_reset when tamper flag is set
tamp_itamp16	Boot ROM fault
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. Content is erased on first write access to backup RAM (so after next POR and first write access to backup RAM if system was in VBAT mode).
2. This monitoring must be enabled by setting MONEN in *PWR control register 2 (PWR_CR2)*.
3. This monitoring must be enabled by setting LSECSSON in *RCC backup domain control register (RCC_BDCR)*.
4. HSE monitoring is an OR between HSECSS output, and LPTIM1 output (lptim_ch1) when it is configured for HSE over-frequency detection. HSECSS must be enabled by HSECSSON bit in *RCC oscillator clock enable set register (RCC_OCENSETR)*.
5. This signal is generated in the TAMP peripheral.

64.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 63.3.3: GPIOs controlled by the RTC and TAMP*.

64.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBD3P bit in the power control peripheral (refer to the PWR power control section). DBD3P bit must be set in order to enable TAMP registers write access.

64.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register (TAMP_SECCFGR), the Backup registers RIF registers (TAMP_BKPRIFRx, x = 1,2,3) and the Resource x CID configuration registers (TAMP_RxCIDCFGR, x = 0 to 2) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR, TAMP_BKPRIFRx(x=1,2,3), TAMP_RxCIDCFGR (x = 0 to 2) is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in

secure mode, except for the backup registers and monotonic counters which have their own protection setting.

- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode.
- When the CNT2SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT2R can be read and written only in secure mode.
- When the BKPRWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV can be written only in secure mode.
- When the BKPWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV and BKPWPRIV can be written only in secure mode.

A nonsecure access to a secure-protected register is denied:

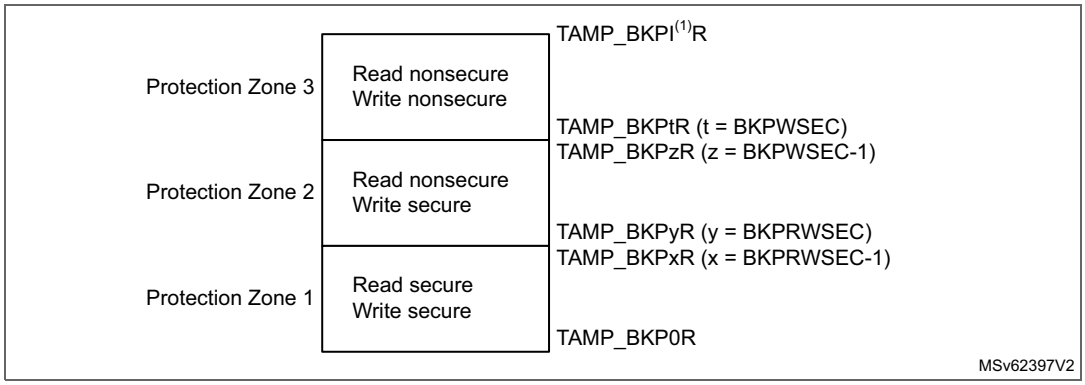
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the IAC (illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

64.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 733. Backup registers protection zones



1. I = last backup register index

64.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and unprivileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR), the Backup registers RIF registers (TAMP_BKPRIFRx, x=1,2,3) and the Resource x CID configuration registers (TAMP_RxCIDCFGR, x = 0 to 2)

which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
  - Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counters which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR, TAMP_BKPRIFRx (x=1,2,3), TAMP_RxCIDCFGR (x=0 to 2) is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- When the CNT2PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT2R can be read and written only in privilege mode.
- When the BKPRWPRIV is set in the TAMP_PRIVCFGR register: the BKPWSEC can be written only in privilege mode.
- When the BKPWPRIV is set in the TAMP_PRIVCFGR register: the BKPRWSEC and BKPWSEC can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 733*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- A notification is generated through a flag/interrupt in the IAC (illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

64.3.8 TAMP compartment-based isolation

By default after a backup domain power-on reset, compartment-based filters are disabled and any compartment has access to the TAMP registers except for the Resource number x CID configuration registers (TAMP_RxCIDCFGR, x = 0 to 2) and the Backup registers RIF registers (TAMP_BKPRIFRx, x=1,2,3) which can be written by the Trusted domain application only (TDCID secure privileged). The CID protection configuration is not affected by a system reset.

Three CID regions are defined in the backup registers, as defined in the *Table 633: CID regions*:

Table 633. CID regions

CID region number	Backup registers area
0	Protection zone 3-RIF0: (Nonsecure read Nonsecure write)

Table 633. CID regions (continued)

CID region number	Backup registers area
1	Protection zone 1-RIF1: (Secure read Secure write) Protection zone 2-RIF1: (Nonsecure read Secure write) Protection zone 3-RIF1: (Nonsecure read Nonsecure write)
2	Protection zone 1-RIF2: (Secure read Secure write) Protection zone 2-RIF2: (Nonsecure read Secure write) Protection zone 3-RIF2: (Nonsecure read Nonsecure write)

Each CID region is allocated to a TAMP resource. The three TAMP CID resources are allocated to TAMP features as shown in *Table 634: RIF TAMP resource number allocation*:

Table 634. RIF TAMP resource number allocation

TAMP resource number	TAMP Feature
0	Tampers and backup registers CID region 0
1	Monotonic counter 1, backup registers CID region 1 and BHK
2	Monotonic counter 2 and backup registers CID region 2

- When CFEN bit is set in the TAMP_R0CIDCFGR:
  - Tampers configuration and status registers: TAMP_CRx (x=1,2,3), TAMP_FLTCR, TAMP_ATCRx(x=1,2), TAMP_ATSEEDR, TAMP_ATOR, TAMP_IER, TAMP_SR, TAMP_MISR, TAMP_SMISR, TAMP_SCR, TAMP_OR and TAMP_RPCFGR can be read and written only by the CID defined in the TAMP_R0CIDCFGR register.
  - Backup registers in Protection zone 3 -RIF0 can be written only by the CID defined in the TAMP_R0CIDCFGR register. They can be read by all CID.
  - TAMPSEC in the TAMP_SECCFGR and TAMPPRIV in the TAMP_PRIVCFGR can be written only by the CID defined in the TAMP_R0CIDCFGR register. They can be read by all CID.
  - BKPRWSEC[7:0] and BKPWSEC[7:0] in the TAMP_SECCFGR, BKPRWPRIV and BKPWPRIV in the TAMP_PRIVCFGR can be written only by TDCID with secure and privilege access only. They can be read by all CID.
- When CFEN bit is set in the TAMP_R1CIDCFGR:
  - TAMP_COUNT1R can be read and written only by the CID defined in the TAMP_R1CIDCFGR register.
  - Backup registers in Protection zone 1-RIF1 can be read and written only by the CID defined in the TAMP_R1CIDCFGR register.
  - Backup registers in Protection zone 2-RIF1 can be written only by the CID defined in the TAMP_R1CIDCFGR register. They can be read only by the CID defined in

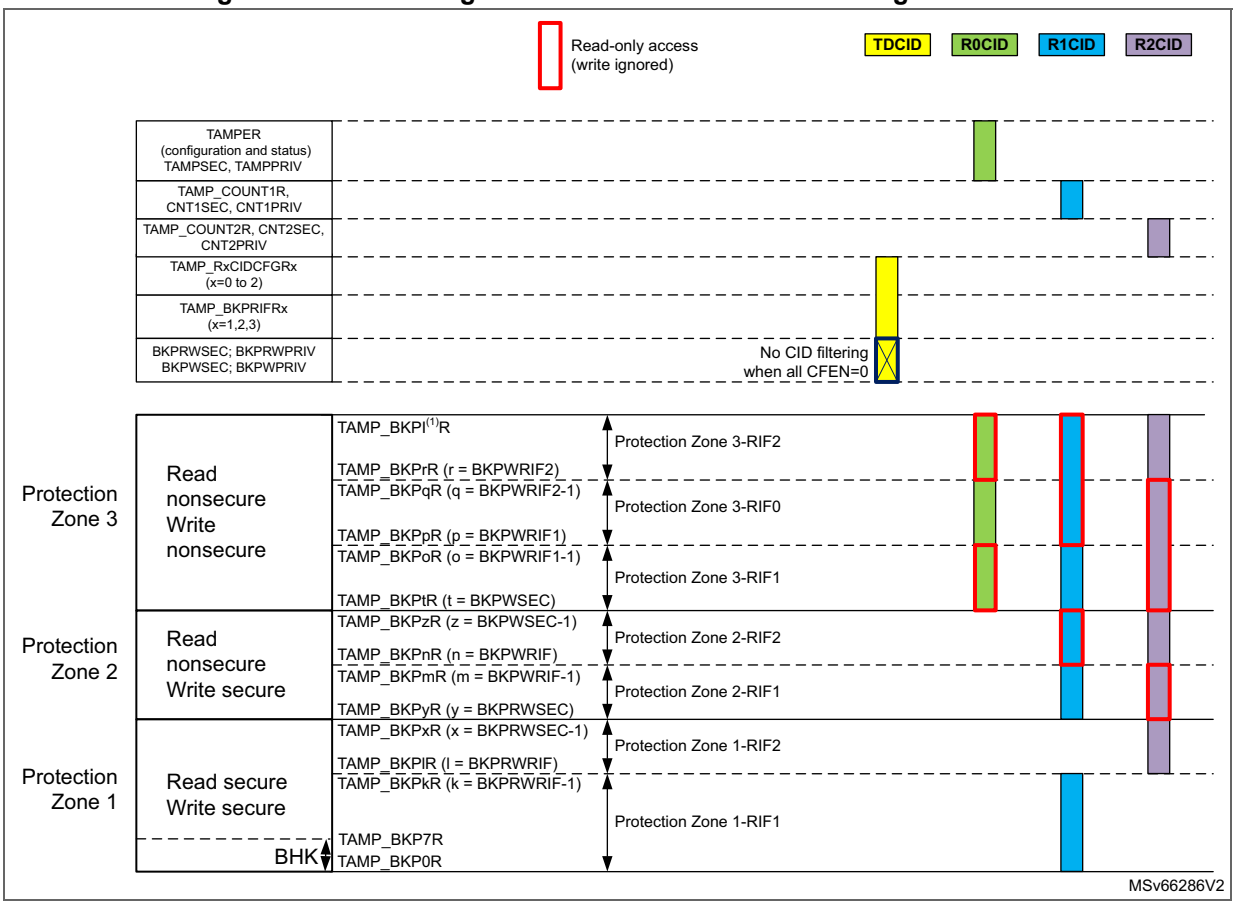
the TAMP_R1CIDCFGR register and by the CID defined in the TAMP_R2CIDCFGR register

- Backup registers in Protection zone 3 -RIF1 can be written only by the CID defined in the TAMP_R1CIDCFGR register. They can be read by all CID.
- BHKLOCK can be written only by the CID defined in the TAMP_R1CIDCFGR register.
- CNT1SEC in the TAMP_SECCFGR and CNT1PRIV in the TAMP_PRIVCFGR can be written only by the CID defined in the TAMP_R1CIDCFGR register. They can be read by all CID.
- BKPRWSEC[7:0] and BKPWSEC[7:0] in the TAMP privilege configuration, BKPRWPRIV and BKPWPRIV in the TAMP_PRIVCFGR can be written only by TDCID with secure and privilege access only. They can be read by all CID.

- When CFEN bit is set in the TAMP_R2CIDCFGR:
  - TAMP_COUNT2R can be read and written only by the CID defined in the TAMP_R2CIDCFGR register.
  - Backup registers in Protection zone 1-RIF2 can be read and written only by the CID defined in the TAMP_R2CIDCFGR register.
  - Backup registers in Protection zone 2-RIF2 can be written only by the CID defined in the TAMP_R2CIDCFGR register. They can be read only by the CID defined in the TAMP_R1CIDCFGR register and by the CID defined in the TAMP_R2CIDCFGR register
  - Backup registers in Protection zone 3 -RIF2 can be written only by the CID defined in the TAMP_R2CIDCFGR register. They can be read by all CID.
  - CNT2SEC in the TAMP_SECCFGR and CNT2PRIV in the TAMP_PRIVCFGR can be written only by the CID defined in the TAMP_R2CIDCFGR register. They can be read by all CID.
  - BKPRWSEC[7:0] and BKPWSEC[7:0] in the TAMP_SECCFGR, BKPRWPRIV and BKPWPRIV in the TAMP_PRIVCFGR can be written only by TDCID with secure and privilege access only. They can be read by all CID.

Caution: The three TAMP_R0CIDCFGR.CFEN, TAMP_R1CIDCFGR.CFEN and TAMP_R2CIDCFGR.CFEN bits must all be set after initialization sequence to protect backup registers with CID filtering. It is not allowed to configure some of the bits to 1 and some others to 0 (in this case the protection would behave as if all these bits were 1).

Figure 734. Access rights for TAMP resources according to the CID



A non-allowed CID access to a CID-filtered register is denied:

- There is no bus error generated.
- A notification is generated through a flag/interrupt in the IAC (illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

64.3.9 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event. It is also cleared with BKERASE command (in all cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

64.3.10 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 632: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 632: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

64.3.11 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 632: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 632: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 632: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 632: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 632: TAMP interconnection*

Table 635. Device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 632: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

64.3.12 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00 and active mode is not selected). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 735. Tamper sampling with precharge pulse

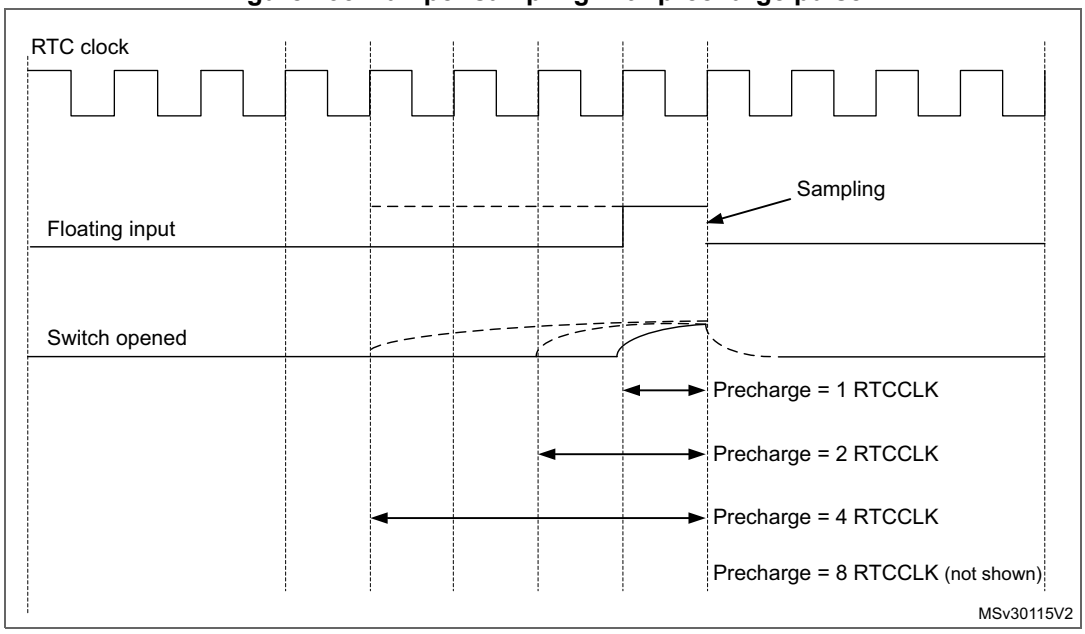
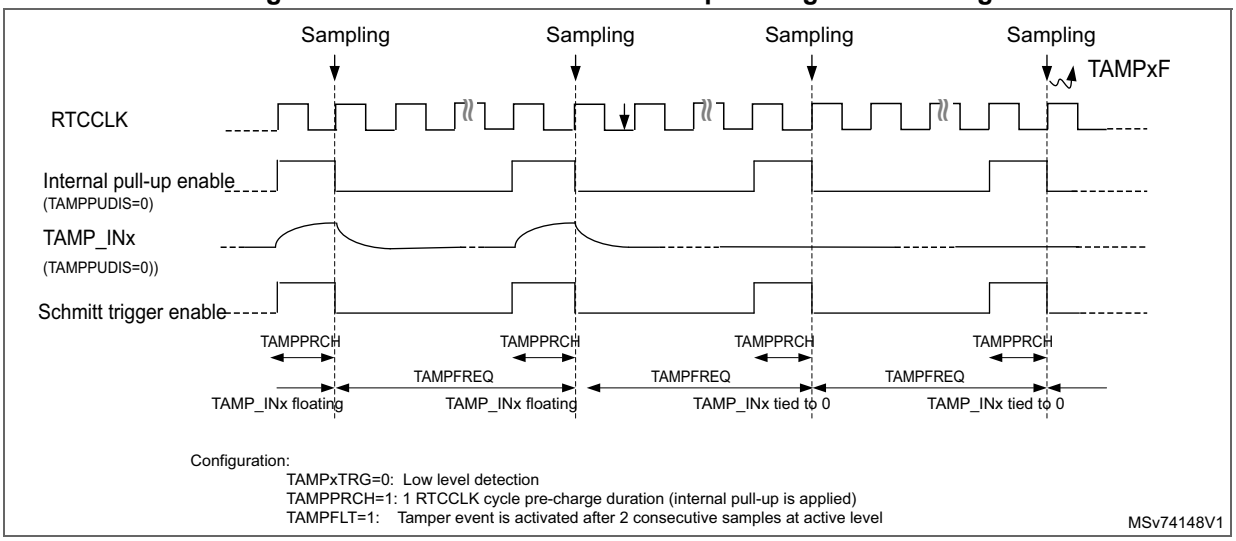


Figure 736. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 636. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…
0x7	$f_{RTCCLK}/128$	0x0	$f_{RTCCLK}/128$	3.906
		0x1	$f_{RTCCLK}/256$	7.8125
		0x2	$f_{RTCCLK}/512$	15.625
		0x3	$f_{RTCCLK}/1024$	31.250
		0x4	$f_{RTCCLK}/2048$	62.5
		0x5	$f_{RTCCLK}/4096$	125
		0x6	$f_{RTCCLK}/8192$	250
		0x7	$f_{RTCCLK}/16384$	500
0x8	$f_{RTCCLK}/256$	0x0	$f_{RTCCLK}/256$	7.8125
		0x1	$f_{RTCCLK}/512$	15.625
		0x2	$f_{RTCCLK}/1024$	31.250
		0x3	$f_{RTCCLK}/2048$	62.5
		0x4	$f_{RTCCLK}/4096$	125
		0x5	$f_{RTCCLK}/8192$	250
		0x6	$f_{RTCCLK}/16384$	500
		0x7	$f_{RTCCLK}/32768$	1000
…	…	…	…	…

Table 636. Active tamper output change period (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0xF	f_{RTCCLK}/32768	0x0	f_{RTCCLK}/32768	1000
		0x1	f_{RTCCLK}/65536	2000
		0x2	f_{RTCCLK}/131072	4000
		0x3	f_{RTCCLK}/262144	8000
		0x4	f_{RTCCLK}/524288	16000
		0x5	f_{RTCCLK}/1048576	32000
		0x6	f_{RTCCLK}/2097152	64000
		0x7	f_{RTCCLK}/4194304	128000

1. Assuming f_{RTCCLK} = 32768 Hz.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 637. Minimum ATPER value

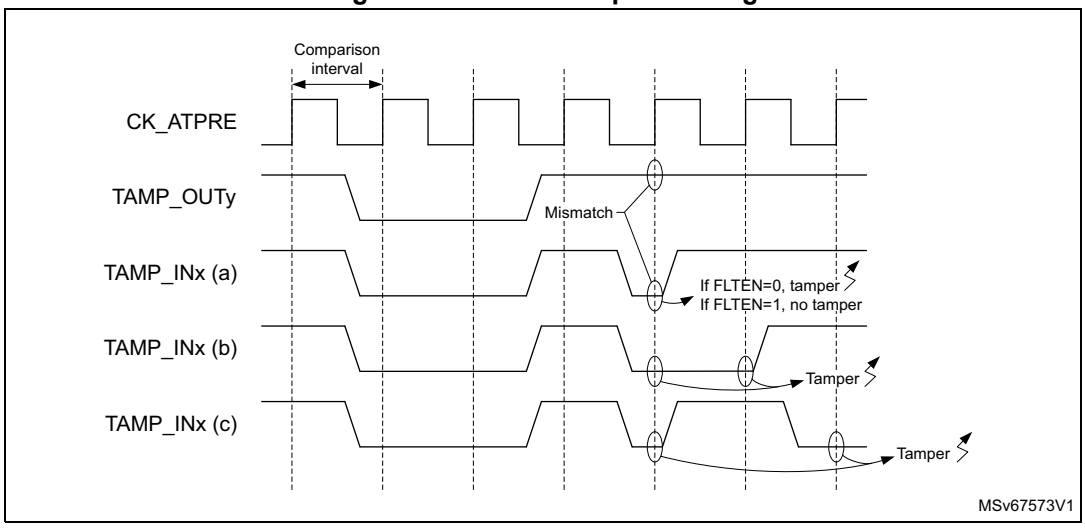
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 737. Active tamper filtering



As illustrated in *Figure 737*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 638* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 638. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0x8	$f_{RTCCLK}/256$	7.812
0x9	$f_{RTCCLK}/512$	15.625
0xA	$f_{RTCCLK}/1024$	31.25
0xB	$f_{RTCCLK}/2048$	62.5

Table 638. Active tamper filtered pulse duration (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0xC	f_{RTCCLK}/4096	125
0xD	f_{RTCCLK}/8192	250
0xE	f_{RTCCLK}/16384	500
0xF	f_{RTCCLK}/32768	1000

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

64.4 TAMP low-power modes

Table 639. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.

Table 640. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[7:1]	Yes	Yes
TAMP_OUT[5:1]	Yes	Yes

64.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).The TAMP provides up to 8 interrupt signals: up to 4 TAMP_CIDz (x = 1 to 4) nonsecure interrupts and up to 4 TAMP_S_CIDz (x = 1 to 4) secure interrupts. Each interrupt signal CIDz can be connected to the corresponding CPU.

TAMP provides an interrupt to the TAMP_CIDz vector only if its configuration is nonsecure and if its CID filter is either disabled, or enabled and configured with CID=z.

TAMP provides an interrupt to the TAMP_S_CIDz vector only if its configuration is secure and if its CID filter is either disabled, or enabled and configured with CID=z.

Table 641. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP_CIDz(z=1 to 4)	Tamper x(2)	TAMPxF	TAMPxIE and TAMPSEC=0 and [CFEN=0 or (CFEN=1 and CID=z)]	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE and TAMPSEC=0 and [CFEN=0 or (CFEN=1 and CID=z)]	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.

2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 639: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

Table 642. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP_S_CIDz (z = 1 to 4)	Tamper x(2)	TAMPxF	TAMPxIE and TAMPSEC = 1 and [CFEN=0 or (CFEN=1 and CID=z)]	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE and TAMPSEC = 1 and [CFEN=0 or (CFEN=1 and CID=z)]	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 639: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

64.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

64.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8:*

TAMP compartment-based isolation.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ITAMP16E	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	ITAMP10E	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **ITAMP16E**: Internal tamper 16 enable
0: Internal tamper 16 disabled.
1: Internal tamper 16 enabled.

Bit 30 **ITAMP15E**: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

Bit 29 **ITAMP14E**: Internal tamper 14 enable
0: Internal tamper 14 disabled.
1: Internal tamper 14 enabled.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 **ITAMP10E**: Internal tamper 10 enable
0: Internal tamper 10 disabled.
1: Internal tamper 10 enabled.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7E**: Tamper detection on TAMP_IN7 enable(1)
0: Tamper detection on TAMP_IN7 is disabled.
1: Tamper detection on TAMP_IN7 is enabled.

Bit 5 **TAMP6E**: Tamper detection on TAMP_IN6 enable(1)
0: Tamper detection on TAMP_IN6 is disabled.
1: Tamper detection on TAMP_IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable(1)
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable(1)
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

64.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TAMP7 TRG	TAMP6 TRG	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
	rw	rw	rw	rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7 POM	TAMP6 POM	TAMP5 POM	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TAMP7TRG**: Active level for tamper 7 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 7 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 7 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 7 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 7 input falling edge triggers a tamper detection event.

Bit 29 **TAMP6TRG**: Active level for tamper 6 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 6 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 6 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 6 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 6 input falling edge triggers a tamper detection event.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 5 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
- 0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input
0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input
0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase
Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked
0: backup registers and device secrets(1) can be accessed if no tamper flag is set
1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask
0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask
0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask
0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7POM**: Tamper 7 potential mode
0: Tamper 7 event detection is in confirmed mode(1).
1: Tamper 7 event detection is in potential mode(2).

Bit 5 **TAMP6POM**: Tamper 6 potential mode
0: Tamper 6 event detection is in confirmed mode(1).
1: Tamper 6 event detection is in potential mode(2).

Bit 4 **TAMP5POM**: Tamper 5 potential mode
0: Tamper 5 event detection is in confirmed mode(1).
1: Tamper 5 event detection is in potential mode(2).

Bit 3 **TAMP4POM**: Tamper 4 potential mode
0: Tamper 4 event detection is in confirmed mode(1).
1: Tamper 4 event detection is in potential mode(2).

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 632: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 632: TAMP interconnection*.

64.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITAMP 16POM	ITAMP 15POM	ITAMP 14POM	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	ITAMP 10POM	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **ITAMP16POM**: Internal tamper 16 potential mode
0: Internal tamper 16 event detection is in confirmed mode(1).
1: Internal tamper 16 event detection is in potential mode(2).

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode(1).
1: Internal tamper 15 event detection is in potential mode(2).

Bit 13 **ITAMP14POM**: Internal tamper 14 potential mode
0: Internal tamper 14 event detection is in confirmed mode(1).
1: Internal tamper 14 event detection is in potential mode(2).

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 **ITAMP10POM**: Internal tamper 10 potential mode
0: Internal tamper 10 event detection is in confirmed mode(1).
1: Internal tamper 10 event detection is in potential mode(2).

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 632: TAMP interconnection*

2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 632: TAMP interconnection*.

64.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

64.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	TAMP7 AM	TAMP6 AM	TAMP5 AM	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 637: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE = RTCCLK/$2^{ATCKSEL}$.

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection

00: TAMPOUTSEL4 = TAMP_OUT1

01: TAMPOUTSEL4 = TAMP_OUT2

10: TAMPOUTSEL4 = TAMP_OUT3

11: TAMPOUTSEL4 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection

00: TAMPOUTSEL3 = TAMP_OUT1

01: TAMPOUTSEL3 = TAMP_OUT2

10: TAMPOUTSEL3 = TAMP_OUT3

11: TAMPOUTSEL3 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11: TAMPOUTSEL2 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11: TAMPOUTSEL1 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7AM**: Tamper 7 active mode
0: Tamper 7 detection mode is passive.
1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode
0: Tamper 6 detection mode is passive.
1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode
0: Tamper 5 detection mode is passive.
1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

64.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEED[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

64.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

64.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ATOSEL7[2:0]			ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2]
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ATOSEL7[2:0]**: Active tamper shared output 7 selection
000: TAMPOUTSEL7 = TAMP_OUT1
001: TAMPOUTSEL7 = TAMP_OUT2
010: TAMPOUTSEL7 = TAMP_OUT3
011: TAMPOUTSEL7 = TAMP_OUT4
100: TAMPOUTSEL7 = TAMP_OUT5
101: TAMPOUTSEL7 = TAMP_OUT6
110: TAMPOUTSEL7 = TAMP_OUT7
111: TAMPOUTSEL7 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 25:23 **ATOSEL6[2:0]**: Active tamper shared output 6 selection
000: TAMPOUTSEL6 = TAMP_OUT1
001: TAMPOUTSEL6 = TAMP_OUT2
010: TAMPOUTSEL6 = TAMP_OUT3
011: TAMPOUTSEL6 = TAMP_OUT4
100: TAMPOUTSEL6 = TAMP_OUT5
101: TAMPOUTSEL6 = TAMP_OUT6
110: TAMPOUTSEL6 = TAMP_OUT7
111: TAMPOUTSEL6 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
000: TAMPOUTSEL5 = TAMP_OUT1
001: TAMPOUTSEL5 = TAMP_OUT2
010: TAMPOUTSEL5 = TAMP_OUT3
011: TAMPOUTSEL5 = TAMP_OUT4
100: TAMPOUTSEL5 = TAMP_OUT5
101: TAMPOUTSEL5 = TAMP_OUT6
110: TAMPOUTSEL5 = TAMP_OUT7
111: TAMPOUTSEL5 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: TAMPOUTSEL4 = TAMP_OUT6
110: TAMPOUTSEL4 = TAMP_OUT7
111: TAMPOUTSEL4 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 18:17 are the mirror of ATOSEL4[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: TAMPOUTSEL3 = TAMP_OUT6
110: TAMPOUTSEL3 = TAMP_OUT7
111: TAMPOUTSEL3 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: TAMPOUTSEL2 = TAMP_OUT6
110: TAMPOUTSEL2 = TAMP_OUT7
111: TAMPOUTSEL2 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 12:11 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: TAMPOUTSEL1 = TAMP_OUT6
110: TAMPOUTSEL1 = TAMP_OUT7
111: TAMPOUTSEL1 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 9:8 are the mirror of ATOSEL1[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 7:0 Reserved, must be kept at reset value.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

64.6.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC, CNT2SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 64.3.7: TAMP privilege protection modes*).

Each bit of this register can be individually write-protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	CNT2SEC	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw	rw							rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 64.3.5: TAMP secure protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event.

0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.

1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset

BKPWSEC value must be from 0 to 128.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):

- if TZEN=1, these backup registers can be written only with secure access.
- They can be read with secure or nonsecure access.

If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).

- They can be read or written with secure or nonsecure access.

If BKPWSEC = 128: there is no protection zone 3.

Refer to *Figure 733: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access.

Note: If BKPWPRIV is set, BKPWSEC[7:0] can be written only in privileged mode.

Note: If CFEN=1 in TAMP_R0CIDCFGR or if CFEN=1 in TAMP_R1CIDCFGR or if CFEN=1 in TAMP_R2CIDCFGR, BKPWSEC[7:0] can be written only by Trusted Domain CID with secure and privilege access.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection

0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.

1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bit 14 **CNT2SEC**: Monotonic counter 2 secure protection

0: Monotonic counter 2 (TAMP_COUNT2R) can be read and written when the APB access is secure or nonsecure.

1: Monotonic counter 2 (TAMP_COUNT2R) can be read and written only when the APB access is secure.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset

BKPRWSEC value must be from 0 to 128.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).

- if TZEN=1, these backup registers can be read and written only with secure access.

If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 733: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.

Note: If BKPRWPRIV or BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

Note: If CFEN=1 in TAMP_R0CIDCFGR or if CFEN=1 in TAMP_R1CIDCFGR or if CFEN=1 in TAMP_R2CIDCFGR, BKPRWSEC[7:0] can be written only by Trusted Domain CID with secure and privilege access.

64.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 64.3.5: TAMP secure protection modes*).

Each bit of this register can be individually write-protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	CNT2 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counters and backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 64.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or unprivileged.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bit 14 **CNT2PRIV**: Monotonic counter 2 privilege protection
0: Monotonic counter 2 (TAMP_COUNT2R) can be read and written when the APB access is privileged or unprivileged.
1: Monotonic counter 2 (TAMP_COUNT2R) can be read and written only when the APB access is privileged.

Bits 13:0 Reserved, must be kept at reset value.

64.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ITAMP16IE	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	ITAMP10IE	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **ITAMP16IE**: Internal tamper 16 interrupt enable
0: Internal tamper 16 interrupt disabled.
1: Internal tamper 16 interrupt enabled.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

Bit 29 **ITAMP14IE**: Internal tamper 14 interrupt enable
0: Internal tamper 14 interrupt disabled.
1: Internal tamper 14 interrupt enabled.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 **ITAMP10IE**: Internal tamper 10 interrupt enable
0: Internal tamper 10 interrupt disabled.
1: Internal tamper 10 interrupt enabled.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7IE**: Tamper 7interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7interrupt enabled.

Bit 5 **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

64.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ITAMP16F	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	ITAMP10F	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
									r	r	r	r	r	r	r

Bit 31 **ITAMP16F**: Internal tamper 16 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 16.

Bit 30 **ITAMP15F**: Internal tamper 15 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 **ITAMP14F**: Internal tamper 14 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 14.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 **ITAMP10F**: Internal tamper 10 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 10.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7F**: TAMP7 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5 **TAMP6F**: TAMP6 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

64.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ITAMP16MF	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	ITAMP10MF	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
									r	r	r	r	r	r	r

Bit 31 **ITAMP16MF**: internal tamper 16 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 16 nonsecure interrupt is raised.

Bit 30 **ITAMP15MF**: internal tamper 15 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 15 nonsecure interrupt is raised.

Bit 29 **ITAMP14MF**: internal tamper 14 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 14 nonsecure interrupt is raised.

Bit 28 **ITAMP13MF**: internal tamper 13 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 **ITAMP10MF**: internal tamper 10 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 10 nonsecure interrupt is raised.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 4 nonsecure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 2 nonsecure interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 1 nonsecure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7MF**: TAMP7 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 7 nonsecure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 6 nonsecure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 5 nonsecure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 4 nonsecure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 3 nonsecure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

64.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ITAMP16MF	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	ITAMP10MF	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
									r	r	r	r	r	r	r

Bit 31 **ITAMP16MF**: internal tamper 16 secure interrupt masked flag
This flag is set by hardware when the internal tamper 16 secure interrupt is raised.

Bit 30 **ITAMP15MF**: internal tamper 15 secure interrupt masked flag
This flag is set by hardware when the internal tamper 15 secure interrupt is raised.

Bit 29 **ITAMP14MF**: internal tamper 14 secure interrupt masked flag
This flag is set by hardware when the internal tamper 14 secure interrupt is raised.

Bit 28 **ITAMP13MF**: internal tamper 13 secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 **ITAMP10MF**: internal tamper 10 secure interrupt masked flag
This flag is set by hardware when the internal tamper 10 secure interrupt is raised.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 secure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: Internal tamper 1 secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7MF**: TAMP7 secure interrupt masked flag
This flag is set by hardware when the tamper 7 secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 secure interrupt masked flag
This flag is set by hardware when the tamper 6 secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 secure interrupt masked flag
This flag is set by hardware when the tamper 5 secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

64.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CITAMP16F	CITAMP15F	CITAMP14F	CITAMP13F	CITAMP12F	CITAMP11F	CITAMP10F	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
									w	w	w	w	w	w	w

Bit 31 **CITAMP16F**: Clear ITAMP16 detection flag
Writing 1 in this bit clears the ITAMP16F bit in the TAMP_SR register.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

Bit 29 **CITAMP14F**: Clear ITAMP14 detection flag
Writing 1 in this bit clears the ITAMP14F bit in the TAMP_SR register.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 **CITAMP10F**: Clear ITAMP10 detection flag
Writing 1 in this bit clears the ITAMP10F bit in the TAMP_SR register.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CTAMP7F**: Clear TAMP7 detection flag
Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5 **CTAMP6F**: Clear TAMP6 detection flag
Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

64.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

64.6.17 TAMP monotonic counter 2 register (TAMP_COUNT2R)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x044

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]**: Monotonic counter

This register is read-only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

64.6.18 TAMP option register (TAMP_OR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSDIS	Res.	Res.	IN1_RMP
												rw			rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BSDIS**: Boundary scan disable
0x0: Boundary scan is enabled
0x1: Boundary scan is disabled

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **IN1_RMP**: TAMP_IN1 mapping
0x0: TAMP_IN1 is mapped on PC4
0x1: TAMP_IN1 is mapped on PI8

64.6.19 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG0
															rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **RPCFG0**: Configurable resource 0 protection(1)
0: Resource 0 is not included in the device secrets protected by TAMP peripheral
1: Resource 0 is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg0 and tamp_potential_rpcfg0 signals in *Table 630: TAMP input/output pins* and *Table 632: TAMP interconnection*.

64.6.20 TAMP Backup registers RIF register 1(TAMP_BKPRIFR1)

This register can be written only by the trusted domain CID when the APB access is secure and privileged.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWRIF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWRIF[7:0]**: Protection zone 1-RIF area offset

Backup registers from TAMP_BKP0R to TAMP_BKPkR (k = BKPRWRIF-1) is the **Protection Zone 1-RIF1**. Protection Zone 1-RIF1 belongs to CID region 1.
Backup registers from TAMP_BKPIR (I = BKPRWRIF) to TAMP_BKPxR (x = BKPRWSEC - 1) is the **Protection Zone 1-RIF2**. Protection Zone 1-RIF2 belongs to CID region 2.

64.6.21 TAMP Backup registers RIF register 2(TAMP_BKPRIFR2)

This register can be written only by the trusted domain CID when the APB access is secure and privileged.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPWRIF[7:0]**: Protection zone 2-RIF area offset

Backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPmR (m = BKPWRIF-1) is the **Protection Zone 2-RIF1**. Protection Zone 2-RIF1 belongs to CID region 1.
Backup registers from TAMP_BKPnR (n = BKPWRIF) to TAMP_BKPzR (z = BKPWSEC-1) is the **Protection Zone 2-RIF2**. Protection Zone 2-RIF2 belongs to CID region 2.

64.6.22 TAMP Backup registers RIF register 3(TAMP_BKPRIFR3)

This register can be written only by the trusted domain CID when the APB access is secure and privileged.

Address offset: 0x78

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWRIF2[7:0]**: Protection zone 3-RIF area offset 2

Backup registers from TAMP_BKPpR (p = BKPWRIF1) to TAMP_BKPqR (q = BKPWRIF2-1) is the **Protection Zone 3-RIF0**. Protection Zone 3-RIF0 belongs to CID region 0.
Backup registers from TAMP_BKPrR (r = BKPWRIF2) to the last is the **Protection Zone 3-RIF2**. Protection Zone 3-RIF2 belongs to CID region 2.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPWRIF1[7:0]**: Protection zone 3-RIF area offset 1

Backup registers from TAMP_BKPtR (t = BKPWSEC) to TAMP_BKPoR (p = BKPWRIF1-1) is the **Protection Zone 3-RIF1**. Protection Zone 3-RIF1 belongs to CID region 1.

64.6.23 TAMP Resource x CID configuration register (TAMP_RxCIDCFGR)

This register is used to configure the compartment ID filtering policy for the TAMP local resource number x. Refer to *Section 63.3.6: RTC compartment-based isolation* for TAMP resources number allocation and registers access conditions.

This register can be written only by the trusted domain CID when the APB access is secure and privileged.

Address offset: 0x80 + 0x4 * x (x=0 to 2)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
									rw	rw	rw				rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **CID[2:0]**: Compartment identifier

When CFEN=1, this field is used by the Trusted domain application to define which compartment has access to the TAMP resource number x.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CFEN**: Compartment ID filter enable

0: Compartment-based filtering is disabled for resource number x.

1: Compartment-based filtering is enabled for resource number x.

64.6.24 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 64.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 64.3.7: TAMP privilege protection modes*.

This register can be protected against non-allowed CID access. Refer to *Section 64.3.8: TAMP compartment-based isolation*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 127)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set.

64.6.25 TAMP hardware configuration register 2 (TAMP_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0131 0104

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUM_DEV_SECRETS[7:0]								CID_WIDTH[3:0]				RIF[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRUST_ZONE[3:0]				OPTIONREG_OUT[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **NUM_DEV_SECRETS[7:0]**:
0: TAMP_RPCFGR not implemented
1: TAMP_RPCFGR implemented with 1 bit
2: TAMP_RPCFGR implemented with 2 bits

Bits 23:20 **CID_WIDTH[3:0]**: CID length
1: CID length is 1 bit
2: CID length is 2 bits
3: CID length is 3 bits
4: CID length is 4 bits

Bits 19:16 **RIF[3:0]**: CID compartment filtering
0x0: CID compartment filtering disable
0x1: CID compartment filtering enable

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TRUST_ZONE[3:0]**: Trust zone
0x0: Trust_zone not supported
0x1: Trust_zone supported

Bits 7:0 **OPTIONREG_OUT[7:0]**:
0: Configuration register not implemented
1: Configuration register bit 0 implemented
2: Configuration register bits 1:0 implemented
...
32: Configuration register bits 31:0 implemented

64.6.26 TAMP hardware configuration register 1 (TAMP_HWCFGR1)

Address offset: 0x3F0

Reset value: 0xFFFF 1880

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_TAMPER[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE_TAMPER[3:0]				TAMPER[3:0]				BACKUP_REGS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **INT_TAMPER[15:0]**:
INT_TAMPER[i] = 0: internal tamper i+1 is not implemented (i from 0 to 15)
INT_TAMPER[i] = 1: internal tamper i+1 is implemented (i from 0 to 15)

Bits 15:12 **ACTIVE_TAMPER[3:0]**:
0x0: Active tamper not supported
0x1: Active tamper supported for all external tamper events

Bits 11:8 **TAMPER[3:0]**:
0x0: 16 external tamper events (input pins) supported
0x1: 1 external tamper event (input pin) supported
0x2: 2 external tamper events (input pins) supported
...
0xF: 15 external tamper events (input pins) supported

Bits 7:0 **BACKUP_REGS[7:0]**:
0: No backup register
1: 1 Backup register implemented
2: 2 Backup registers implemented
...
128: 128 backup registers implemented

64.6.27 TAMP version register (TAMP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0035

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

64.6.28 TAMP identification register (TAMP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 1033

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier.

64.6.29 TAMP size identification register (TAMP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier.

64.6.30 TAMP register map

Table 643. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	ITAMP16E	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	ITAMP10E	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x04	TAMP_CR2	Res.	TAMP7TRG	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7POM	TAMP6POM	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value		0	0	0	0	0	0	0	0	0				0	0	0										0	0	0	0	0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP16POM	ITAMP15POM	ITAMP14POM	ITAMP13POM	ITAMP12POM	ITAMP11POM	ITAMP10POM	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	TAMP7AM	TAMP6AM	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	Res.	Res.	Res.	ATO SEL7 [2:0]			ATO SEL6 [2:0]			ATO SEL5 [2:0]			ATO SEL4 [2:0]				ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x20	TAMP_SEC CFGR	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	CNT2SEC	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
	Reset value	0	0							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	CNT2SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0														

Table 643. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TAMP_IER	ITAMP16IE	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	ITAMP10IE	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x30	TAMP_SR	ITAMP16F	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	ITAMP10F	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x34	TAMP_MISR	ITAMP16MF	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	ITAMP10MF	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x38	TAMP_SMISR	ITAMP16MF	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	ITAMP10MF	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x3C	TAMP_SCR	CITAMP16F	CITAMP15F	CITAMP14F	CITAMP13F	CITAMP12F	CITAMP11F	CITAMP10F	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x40	TAMP_ COUNT1R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TAMP_ COUNT2R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSDIS	Res.	Res.	IN1_RMP
	Reset value																													0			0
0x54	TAMP_ RPCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG0
	Reset value																																0
0x70	TAMP_BKPRIFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWRIF[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x74	TAMP_BKPRIFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x78	TAMP_BKPRIFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWRIF1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x80+ 0x04*x,(x=0 to 2)	TAMP_RxCIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CID[2:0]			Res.	Res.	Res.	CFEN
	Reset value																										0	0	0				0

Table 643. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100 + 0x04*x, (x= 0 to 127)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	TAMP_ HWCFGR2	NUM_DEV_SECRETS[7:0]								CID_WIDTH [3:0]				RIF[3:0]				Res.	Res.	Res.	Res.	TRUST_ ZONE[3:0]				OPTIONREG_OUT[7:0]							
	Reset value	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1					0	0	0	1	0	0	0	0	0	1	0	0
0x3F0	TAMP_ HWCFGR1	INT_TAMPER[15:0]																ACTIVE_ TAMPER [3:0]				TAMPER [3:0]				BACKUP_REGS [7:0]							
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0
0x3F4	TAMP_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	1	0	1	0	1
0x3F8	TAMP_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1
0x3FC	TAMP_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

65 Inter-integrated circuit interface (I2C)

65.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

65.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode
- Autonomous functionality in Stop mode

For information on I2C instantiation, refer to *Section 65.3: I2C implementation*.

65.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 644. I2C implementation

I2C features(1)	I2C1	I2C2	I2C3
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
Autonomous in Stop 1/2, LP-Stop 1/2 mode with wake-up capability	X	X	X
SMBus/PMBus	X	X	X

1. X = supported.

65.4 I2C functional description

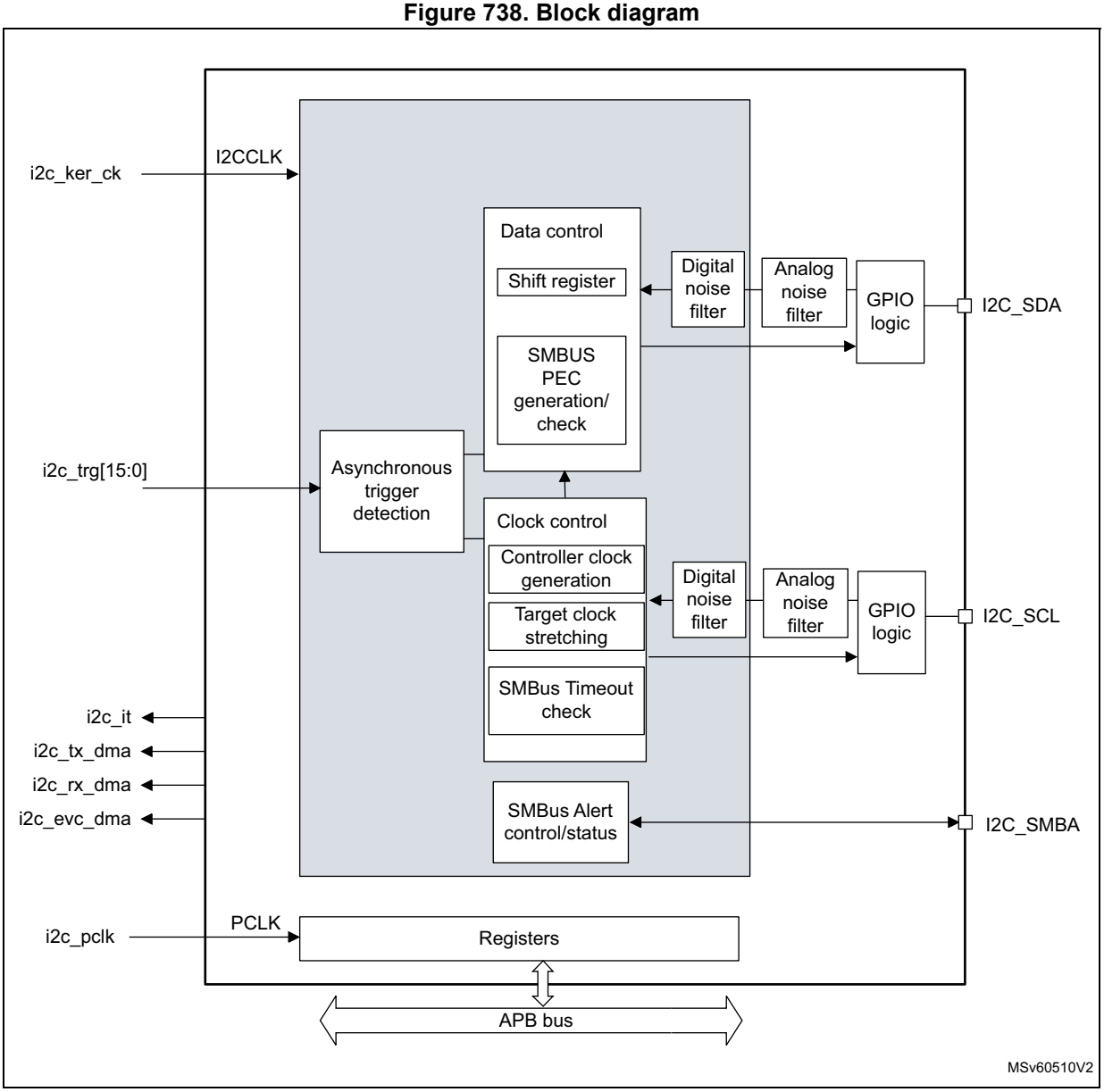
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

65.4.1 I2C block diagram

Figure 738. Block diagram



65.4.2 I2C pins and internal signals

Table 645. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 646. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_trg[15:0]	Input	I2C triggers
i2c_it	Output	I2C interrupts, refer to *Table 662* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)
i2c_evc_dma	Output	I2C event control DMA request (I2C_EVC)

Table 647. I2C1 interconnection

Signal name	Source/destination
i2c_trg0	-
i2c_trg1	-
i2c_trg2	-
i2c_trg3	-
i2c_trg4	exti4(1)
i2c_trg5	exti5(1)
i2c_trg6	-
i2c_trg7	lptim3_ch1
i2c_trg8	lptim4_ch1
i2c_trg9	lptim5_out
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12	exti6(2)
i2c_trg13	exti7(2)
i2c_trg14	lptim1_ch1
i2c_trg15	lptim2_ch1

1. EXTI2 GPIO output trigger.
2. EXTI1 GPIO output trigger.

Table 648. I2C2 interconnection

Signal name	Source/destination
i2c_trg0	-
i2c_trg1	-
i2c_trg2	-
i2c_trg3	-

Table 648. I2C2 interconnection (continued)

Signal name	Source/destination
i2c_trg4	exti4(1)
i2c_trg5	exti5(1)
i2c_trg6	-
i2c_trg7	lptim3_ch1
i2c_trg8	lptim4_ch1
i2c_trg9	lptim5_out
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12	exti6(2)
i2c_trg13	exti7(2)
i2c_trg14	lptim1_ch1
i2c_trg15	lptim2_ch1

1. EXTI2 GPIO output trigger.
2. EXTI1 GPIO output trigger.

Table 649. I2C3 interconnection

Signal name	Source/destination
i2c_trg0	-
i2c_trg1	-
i2c_trg2	-
i2c_trg3	-
i2c_trg4	exti4(1)
i2c_trg5	exti5(1)
i2c_trg6	-
i2c_trg7	lptim3_ch1
i2c_trg8	lptim4_ch1
i2c_trg9	lptim5_out
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12	-
i2c_trg13	-
i2c_trg14	-
i2c_trg15	-

1. EXTI2 GPIO output trigger.

65.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

65.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

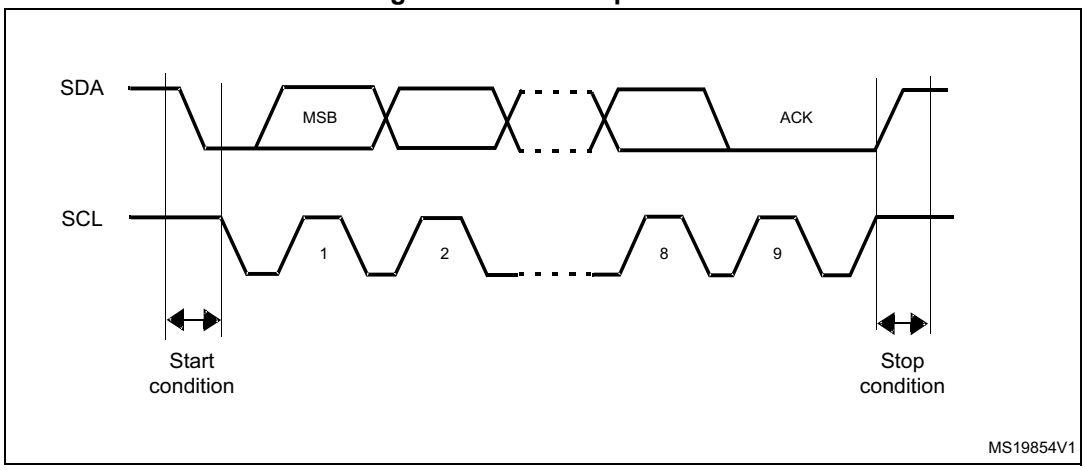
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 739. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

65.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 65.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 650. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Functionality in Stop mode not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

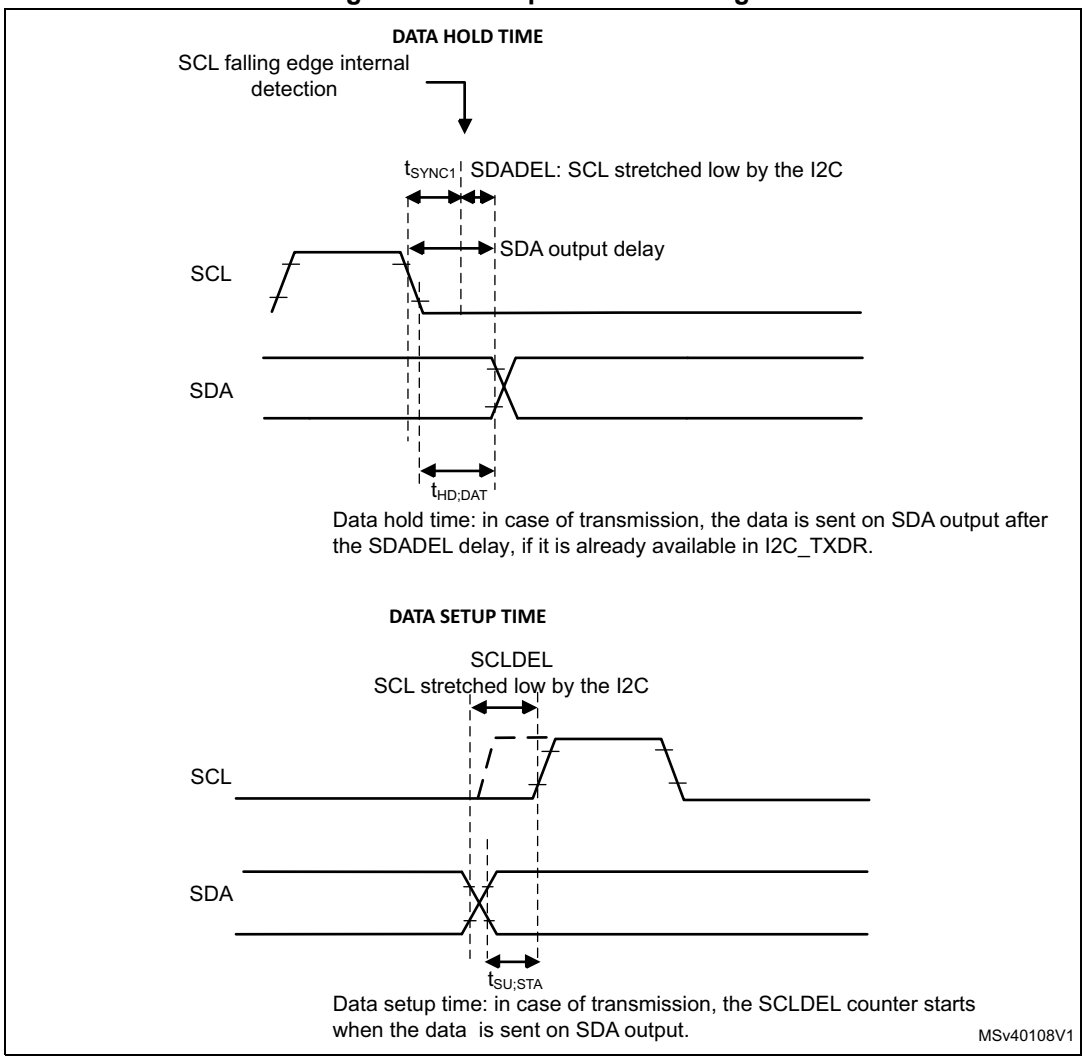
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 740. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\,(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 651. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

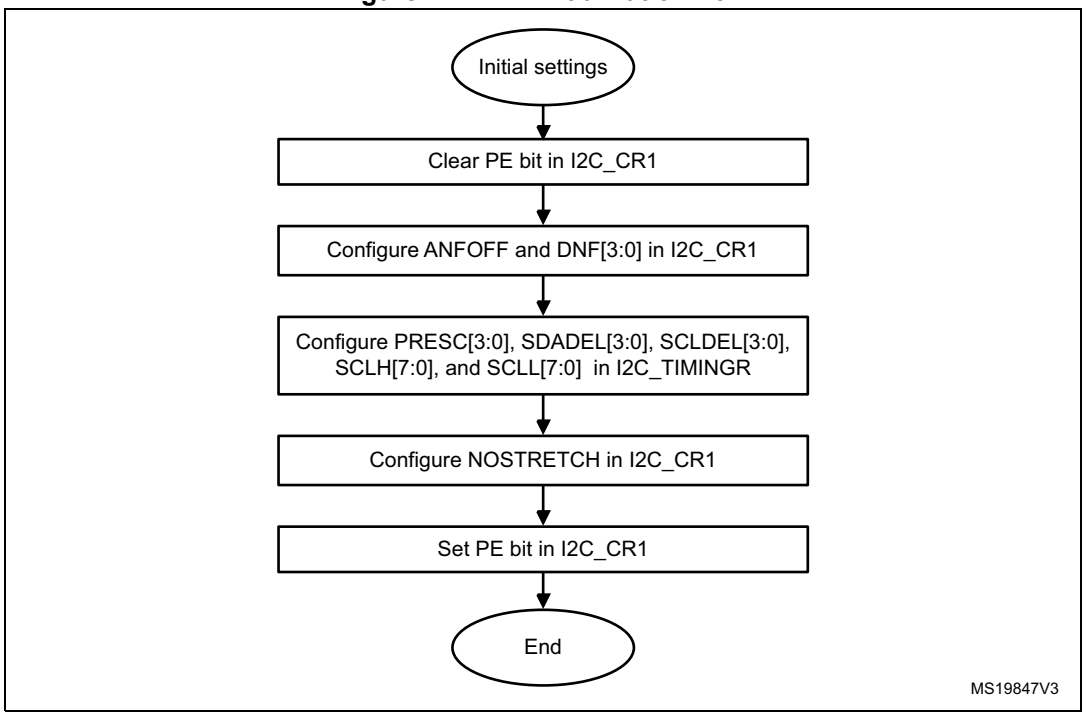
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 741. I2C initialization flow



65.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

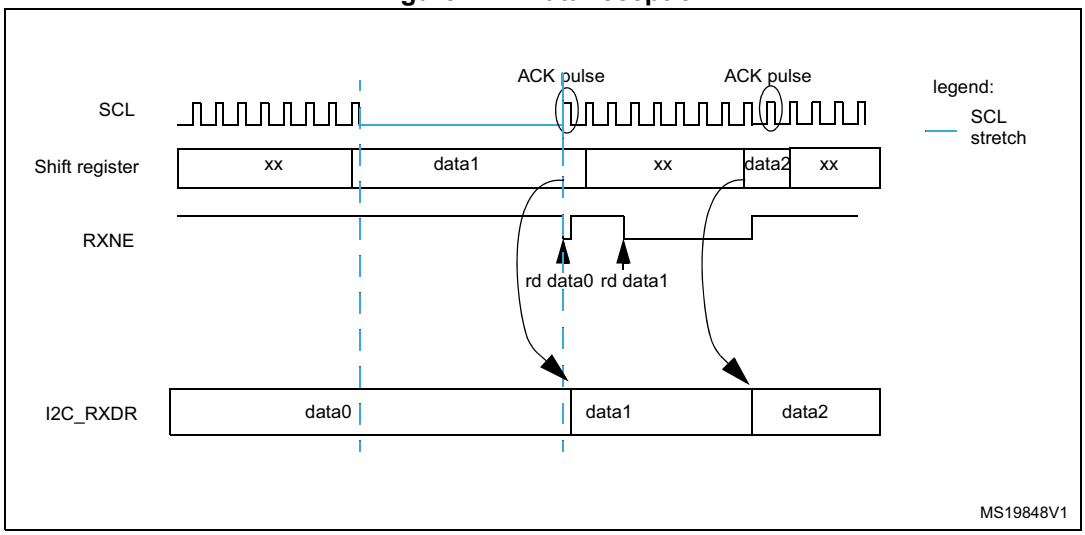
65.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

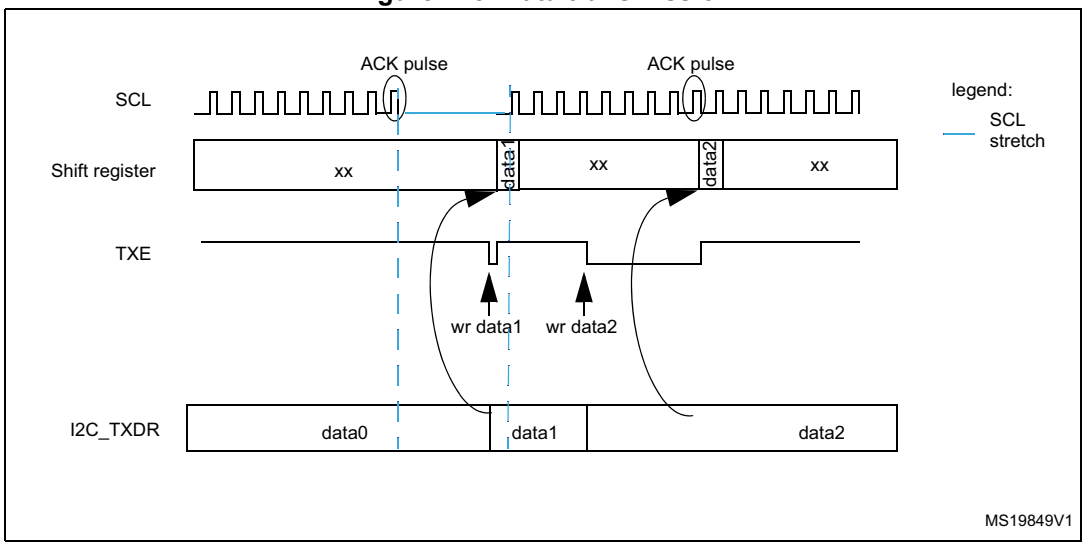
Figure 742. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 743. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 652. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

65.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

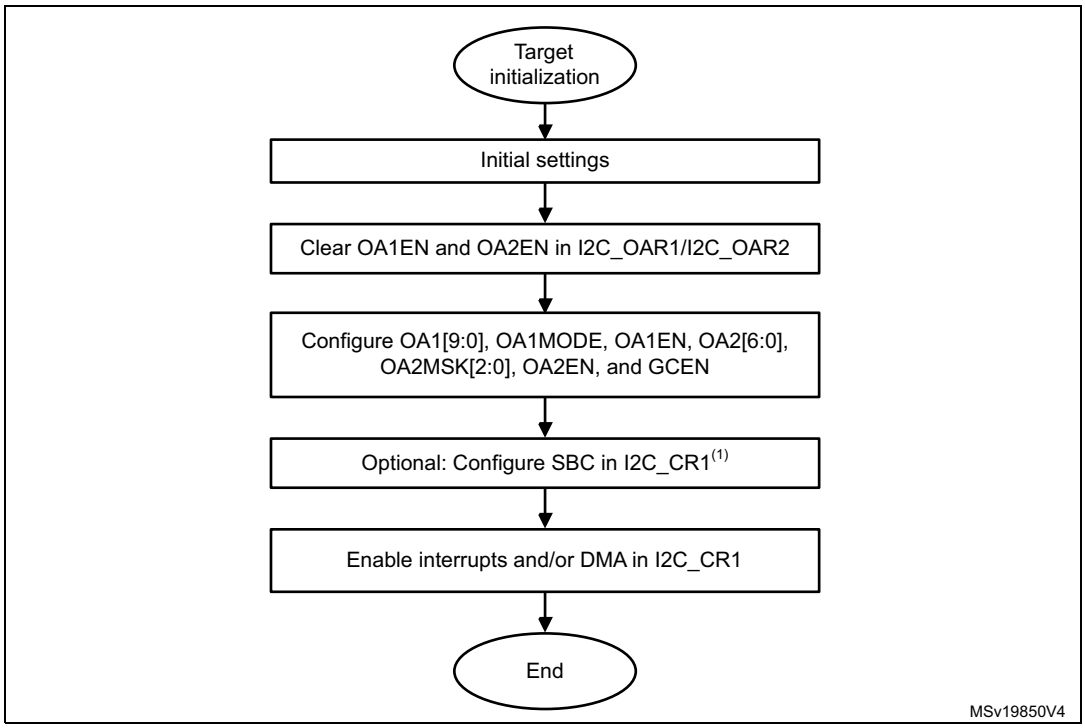
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 744. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 745. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

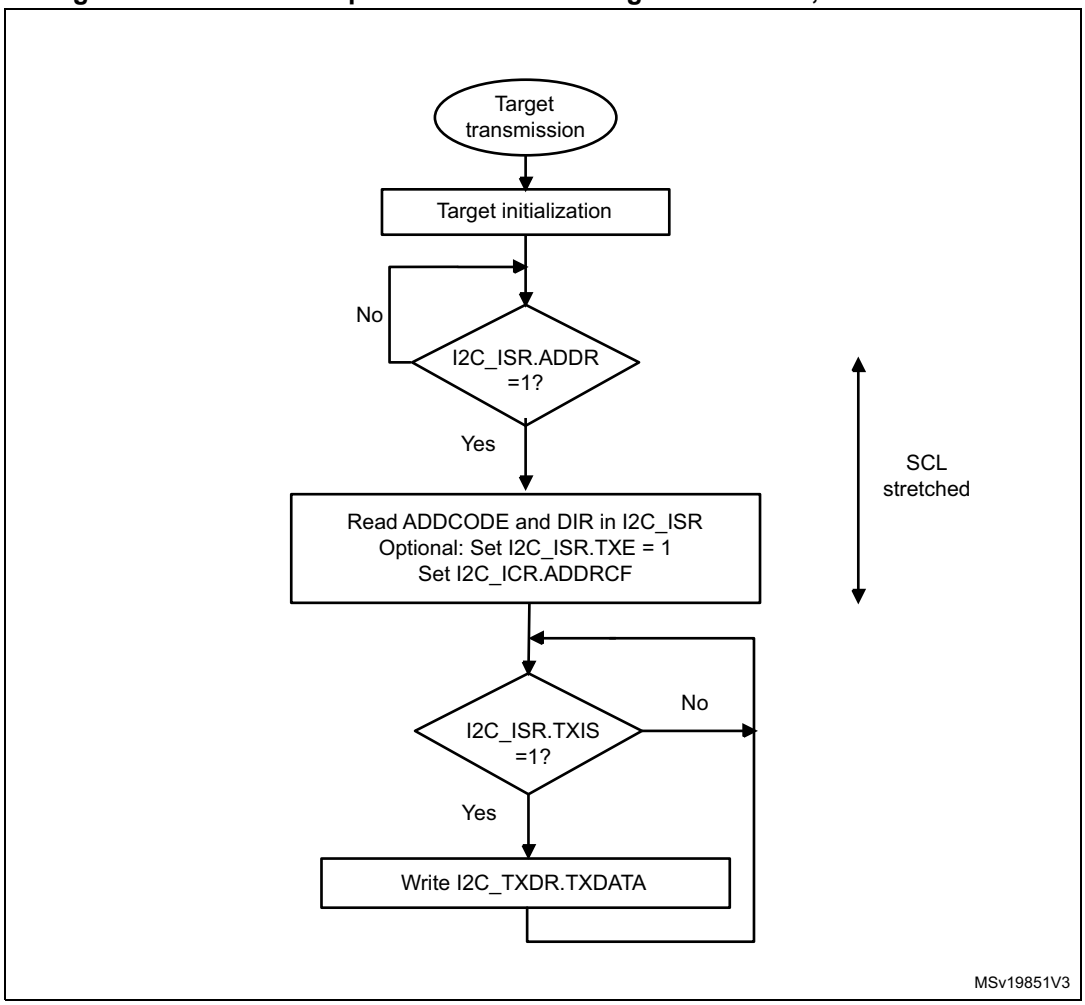


Figure 746. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

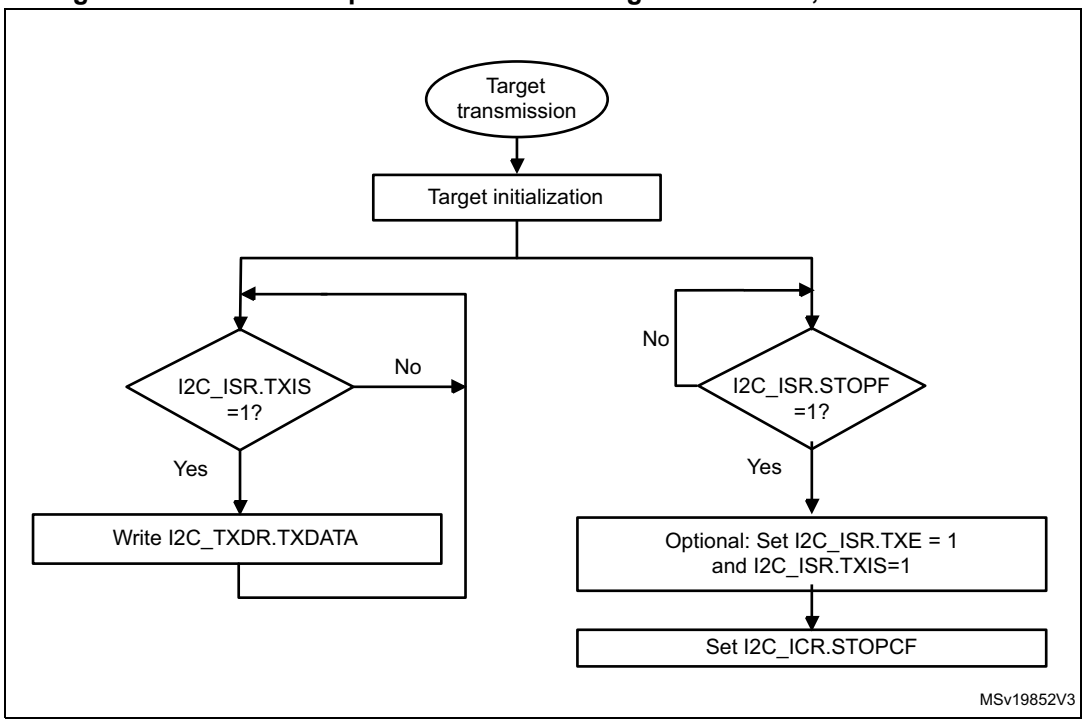
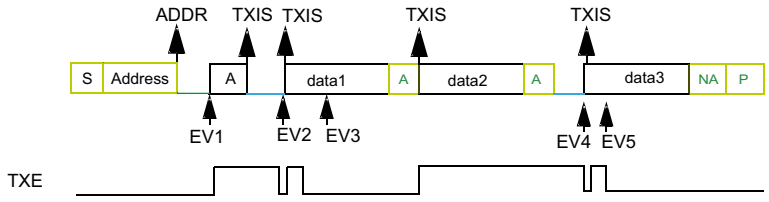


Figure 747. Transfer bus diagrams for I2C target transmitter (mandatory events only)

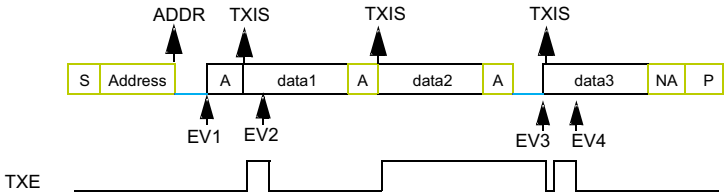
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

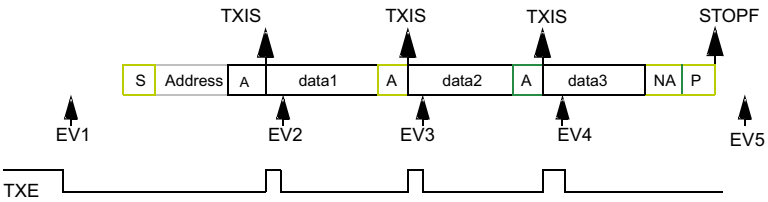
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 748. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

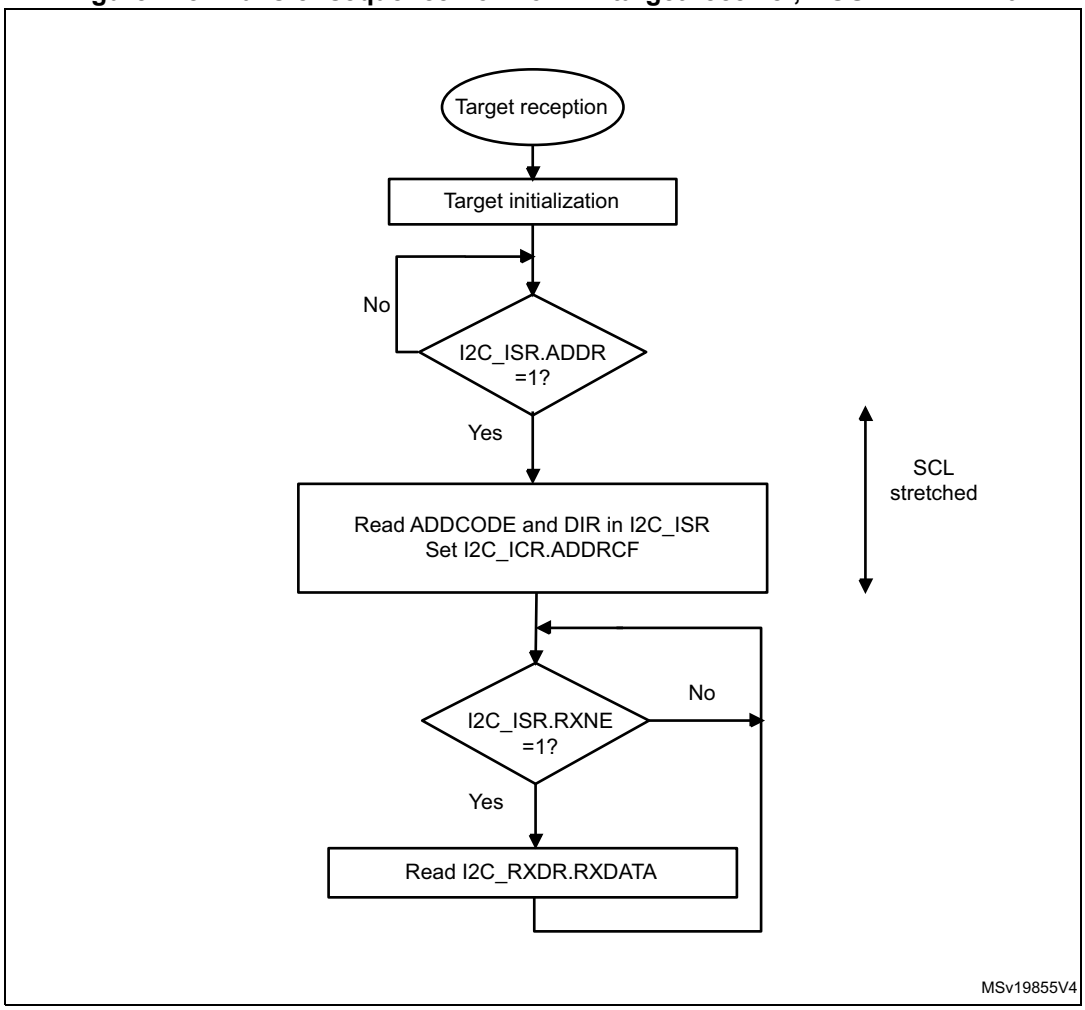


Figure 749. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

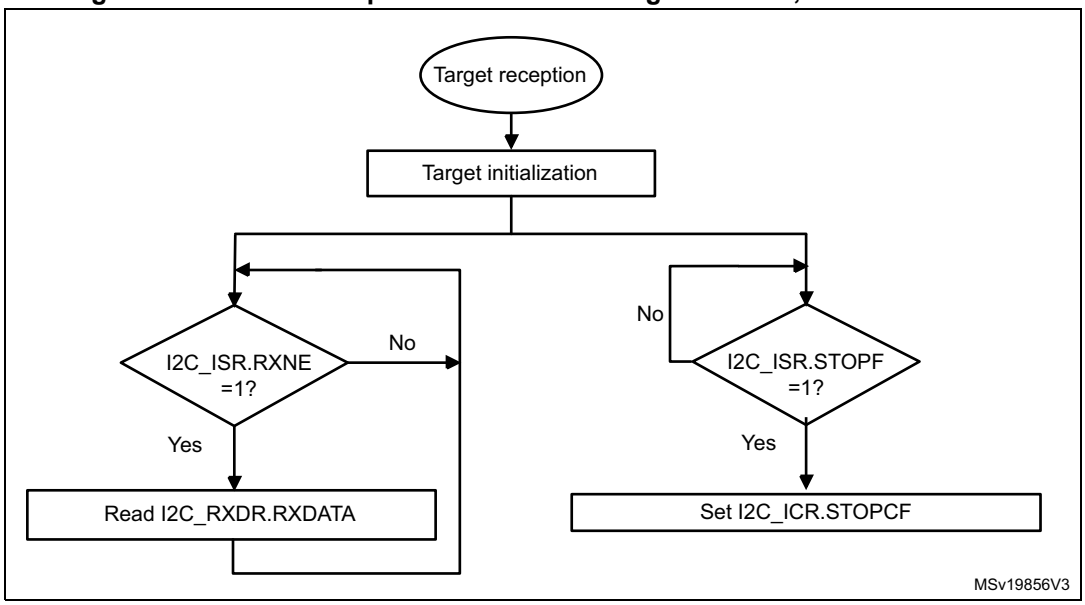
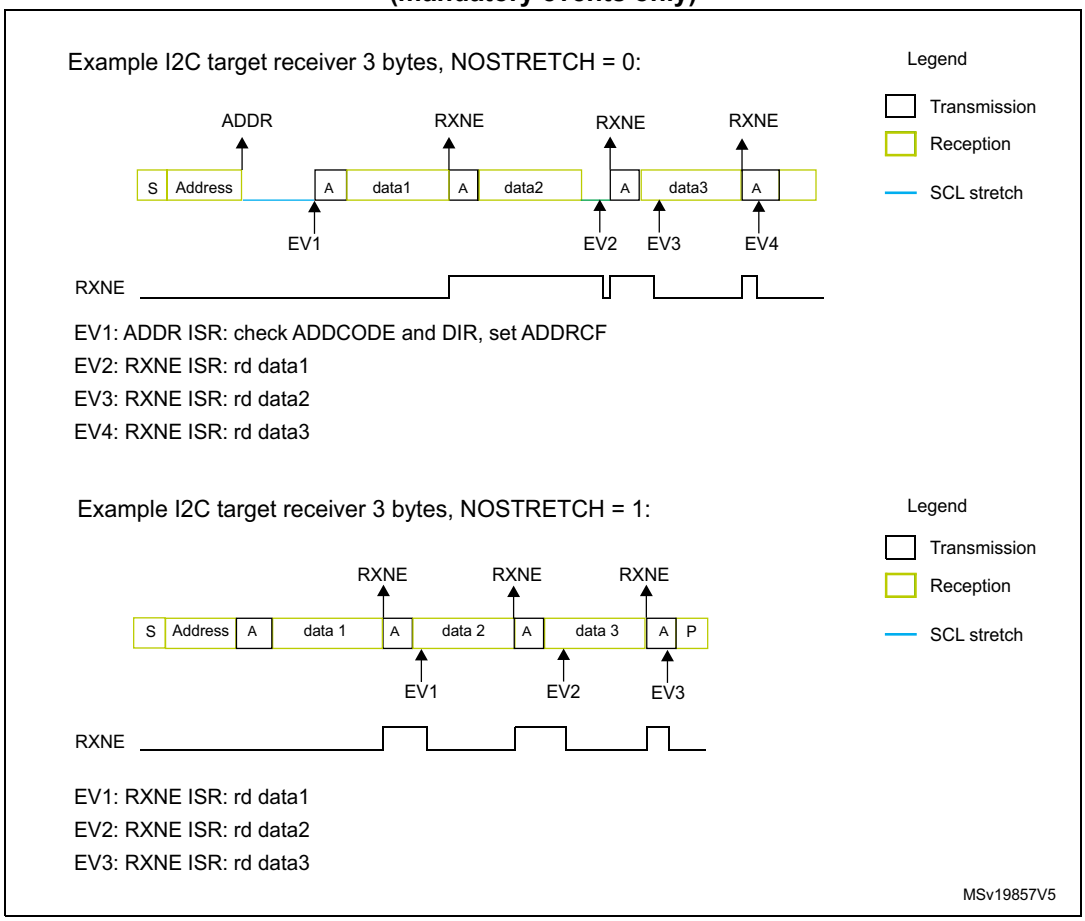


Figure 750. Transfer bus diagrams for I2C target receiver (mandatory events only)



65.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

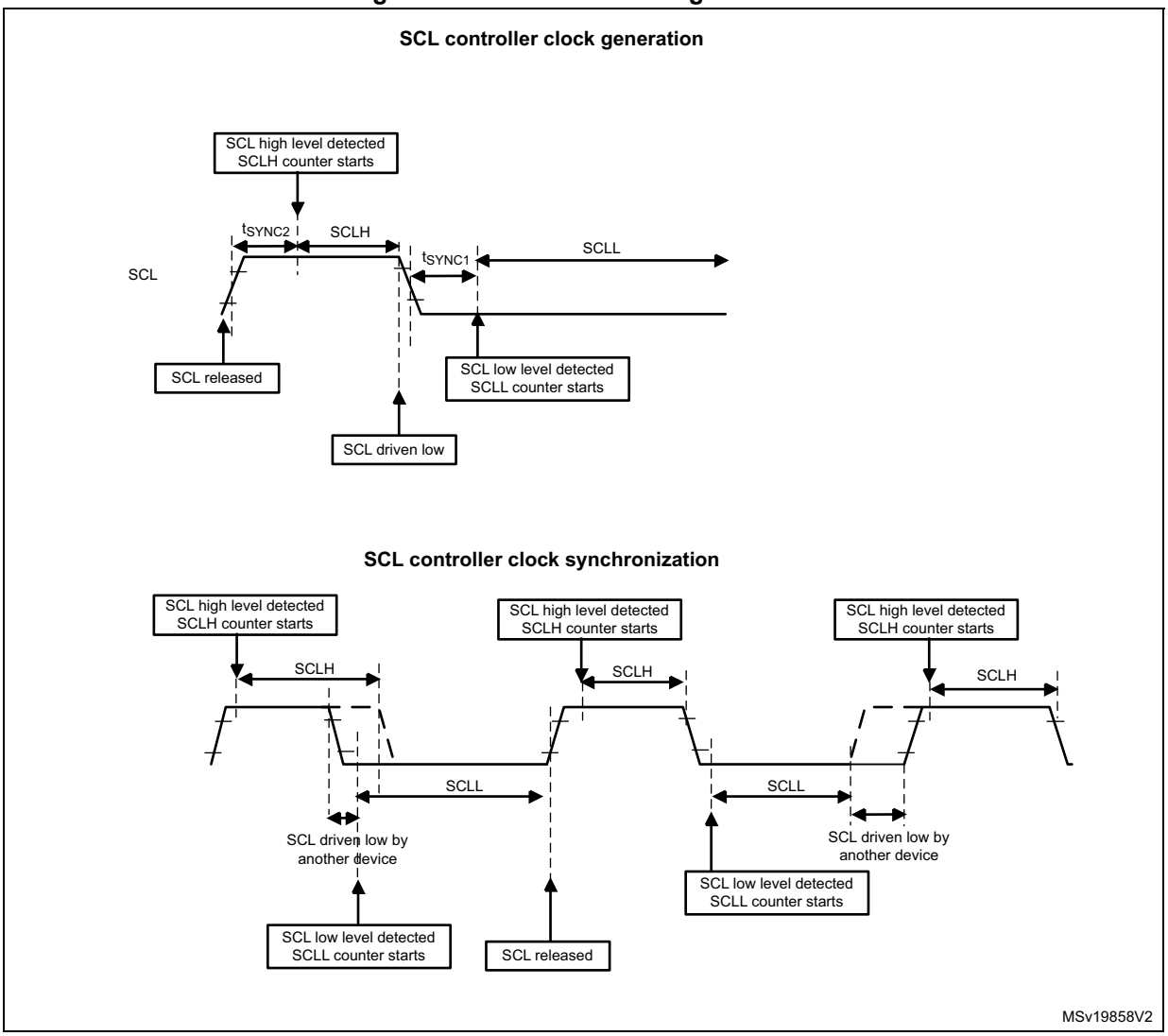
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 751. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 653. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 65.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

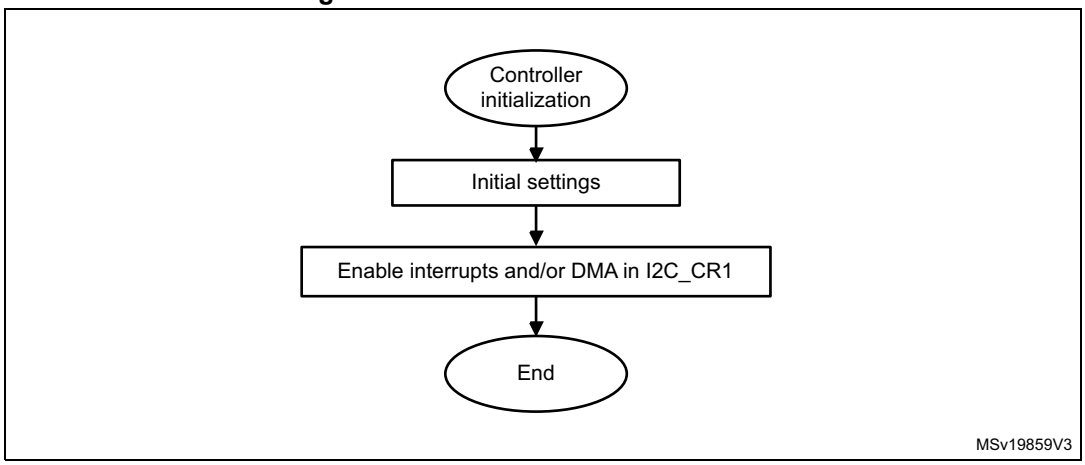
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 752. Controller initialization flow

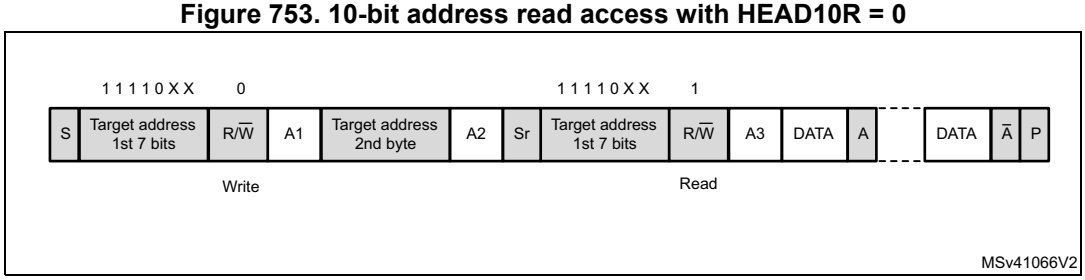


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

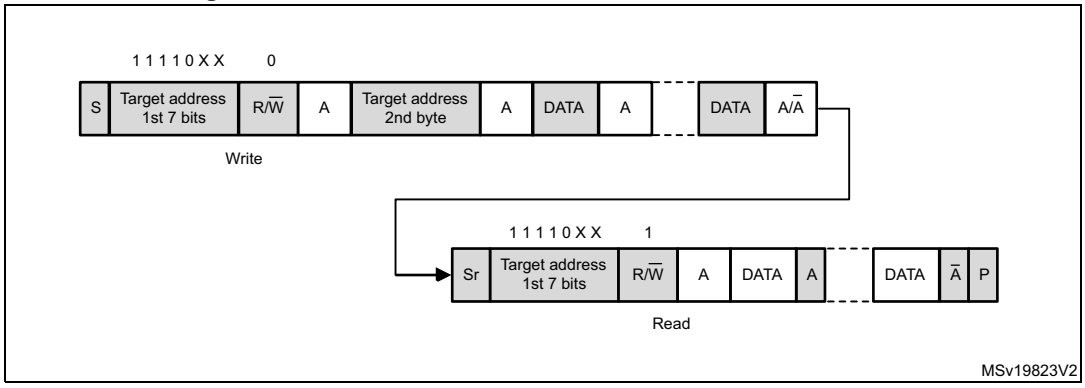
Figure 753. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 754. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 755. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

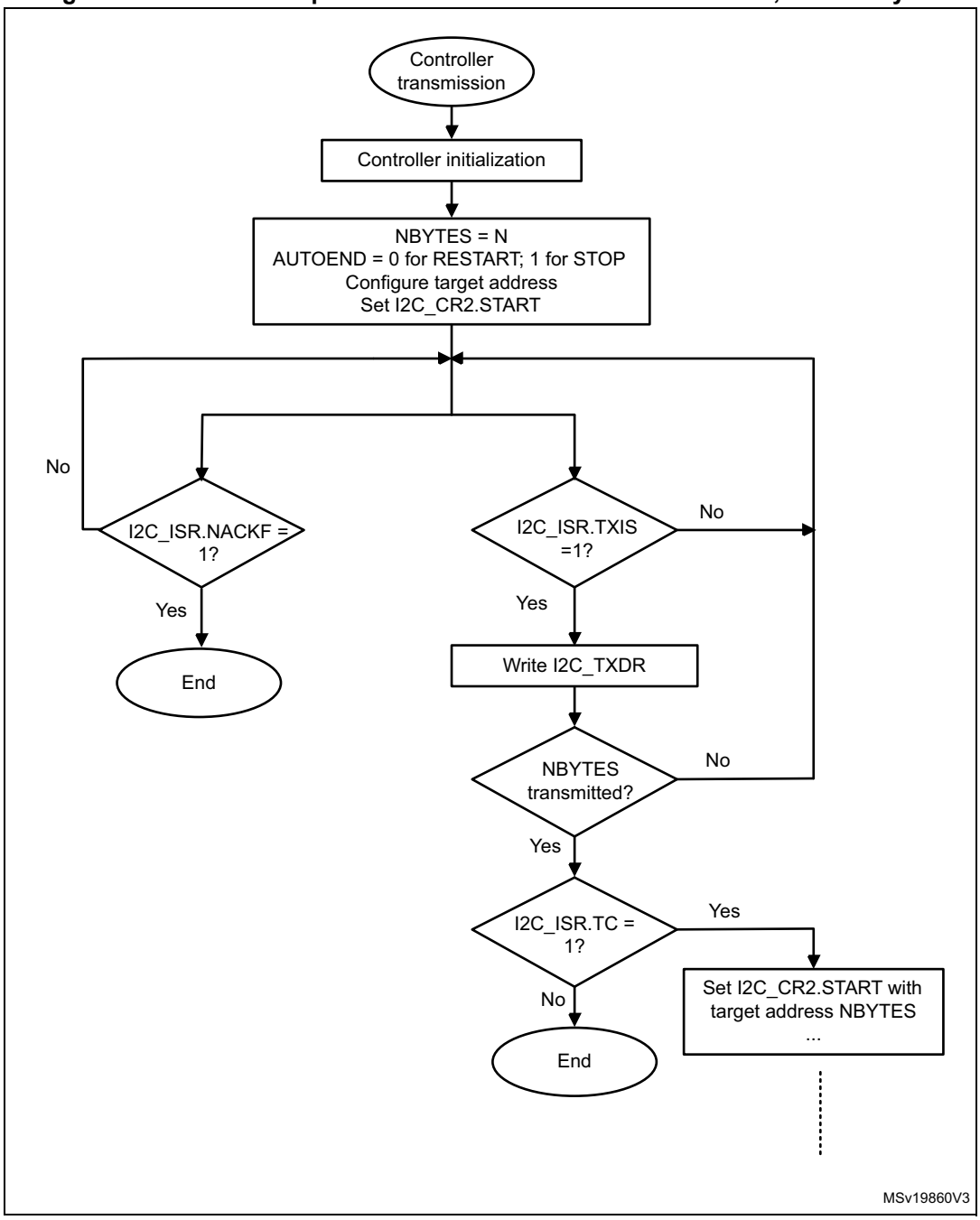


Figure 756. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

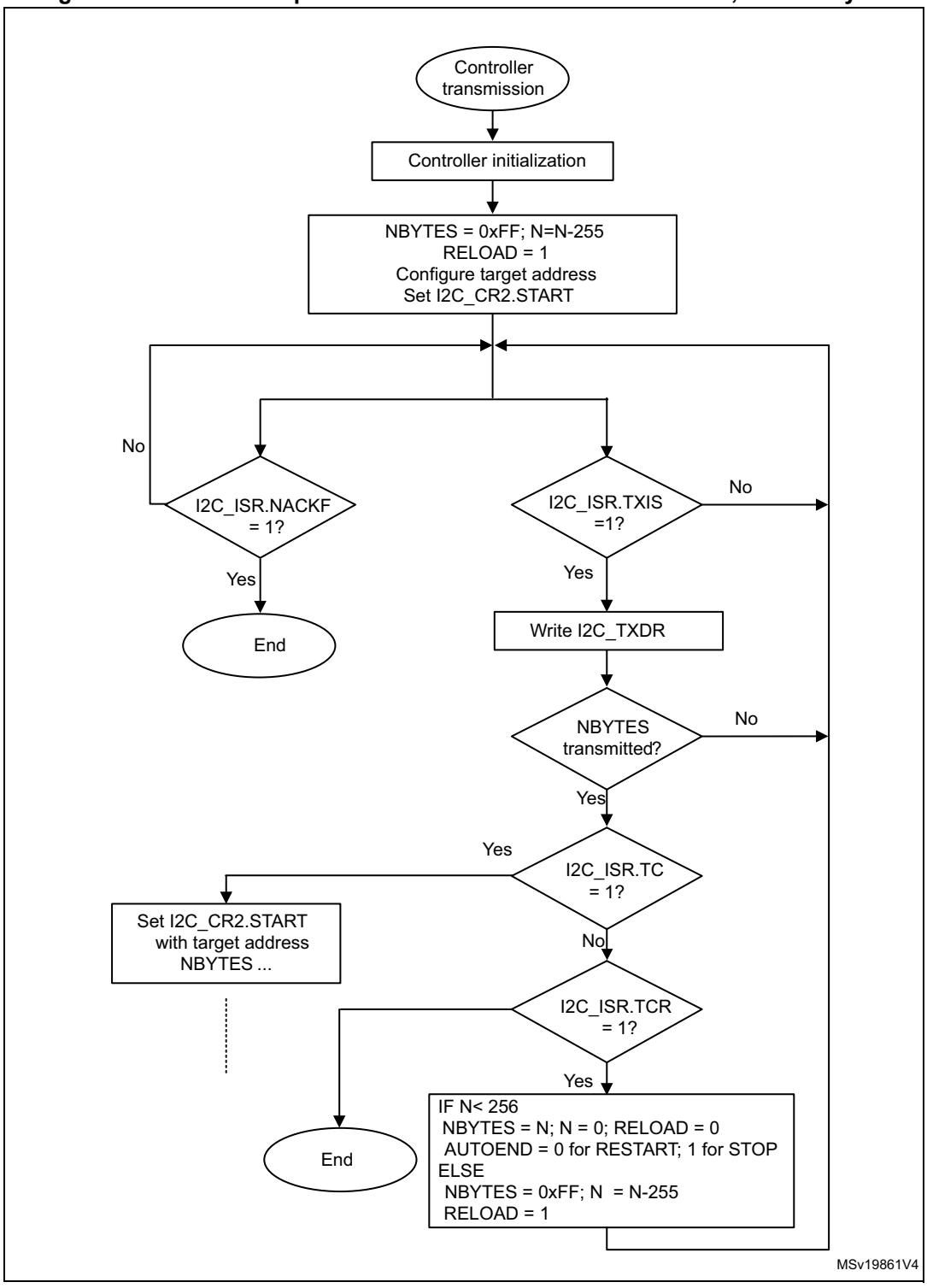
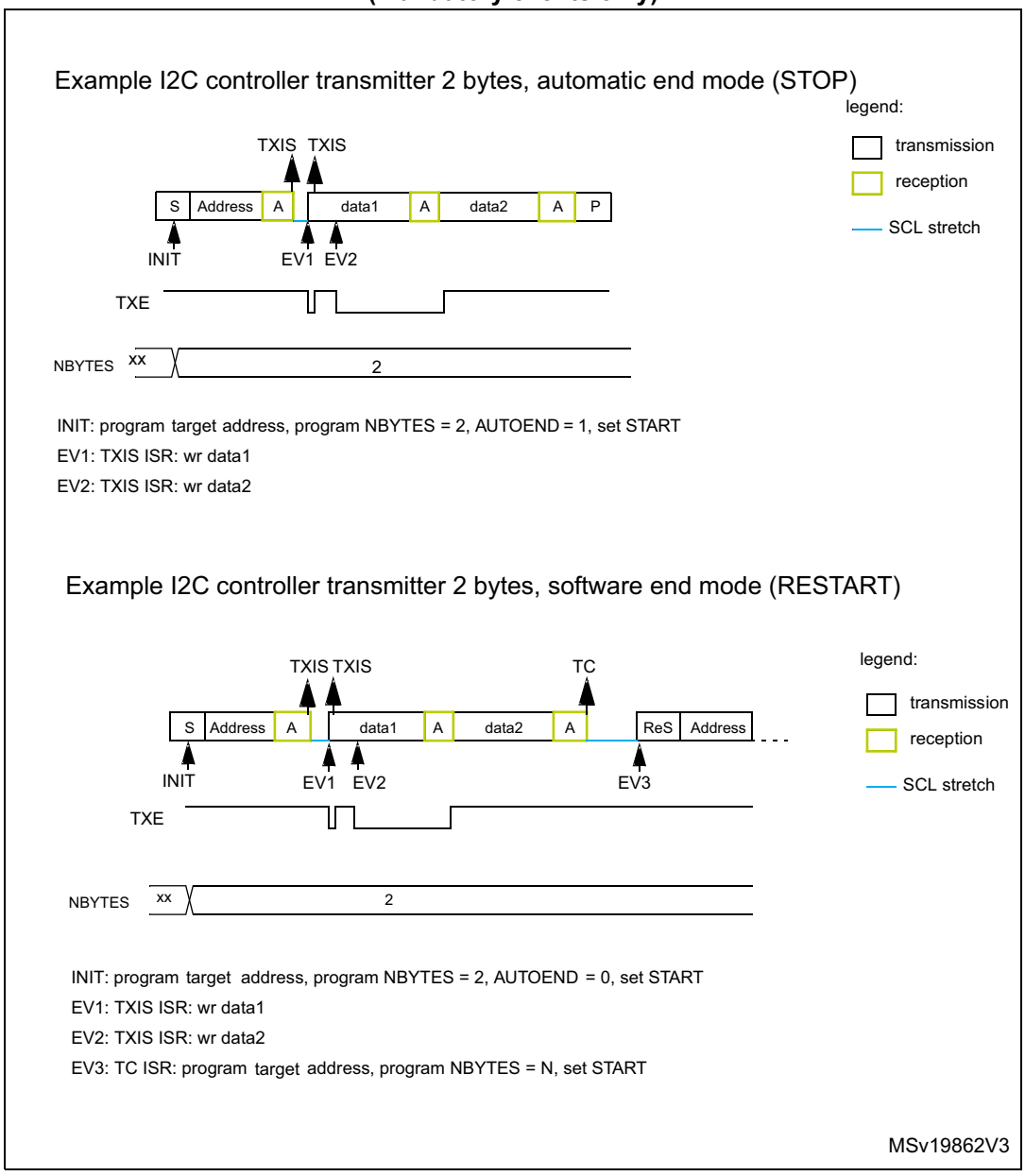


Figure 757. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 758. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

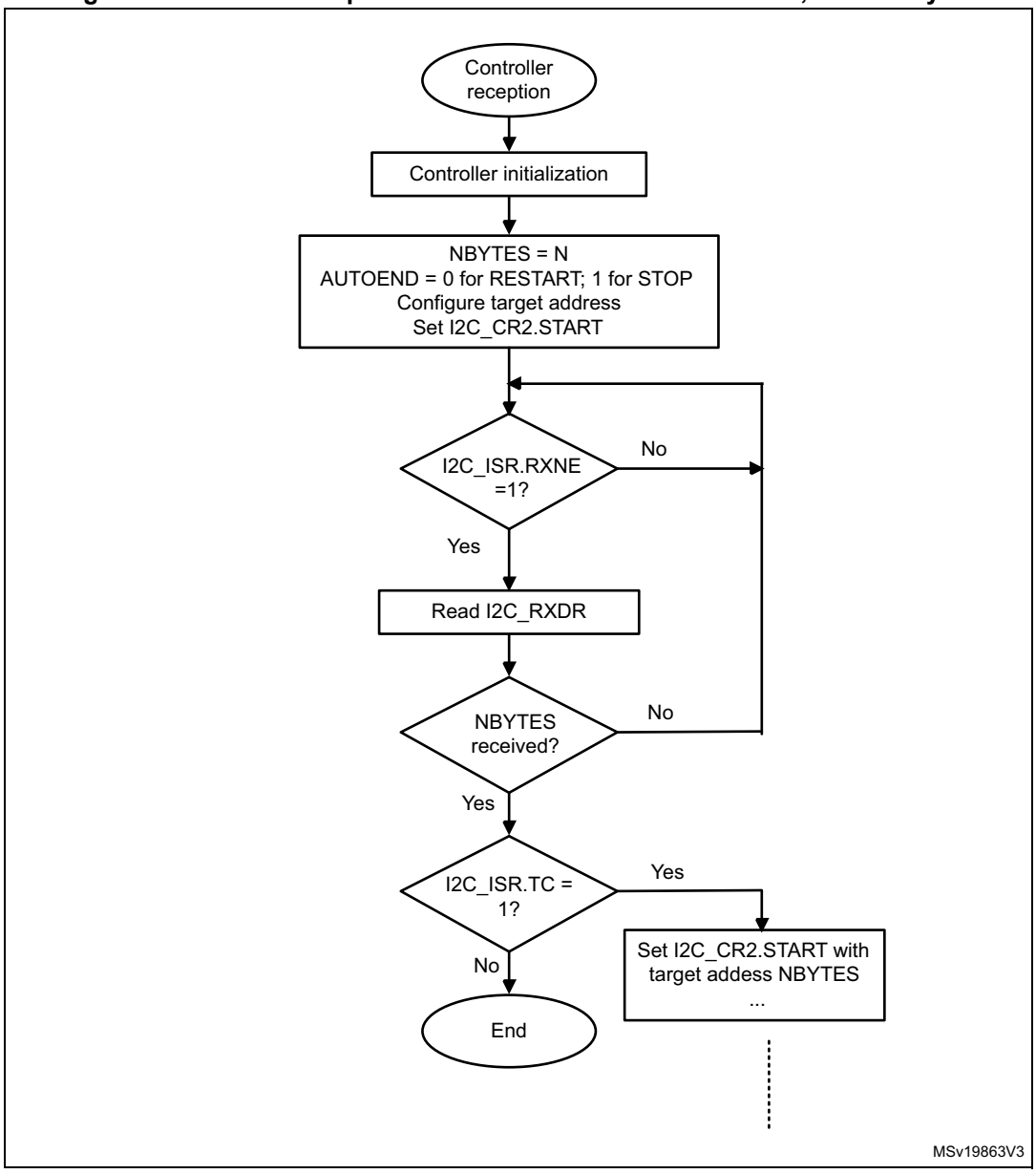


Figure 759. Transfer sequence flow for I2C controller receiver, N > 255 bytes

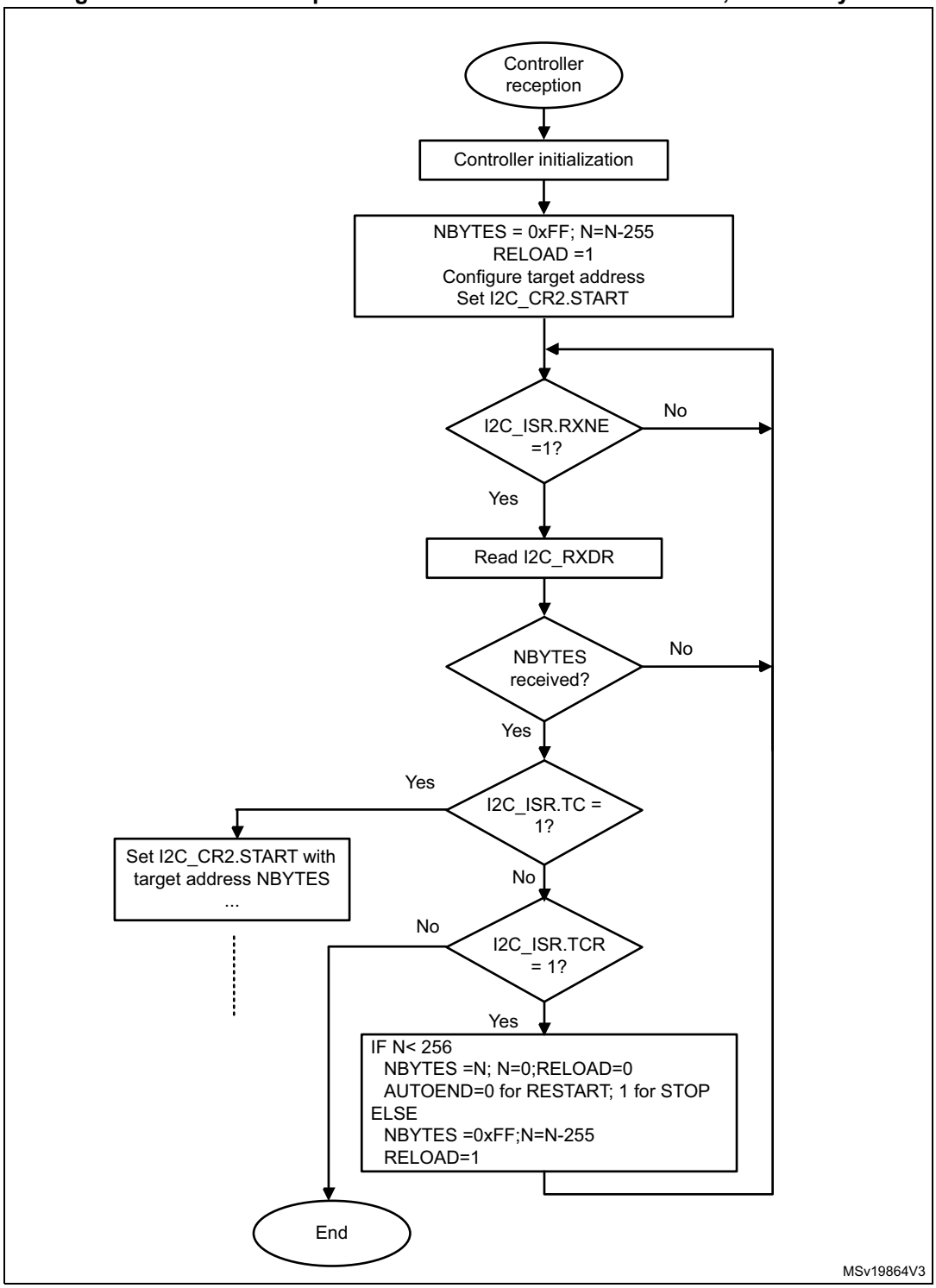
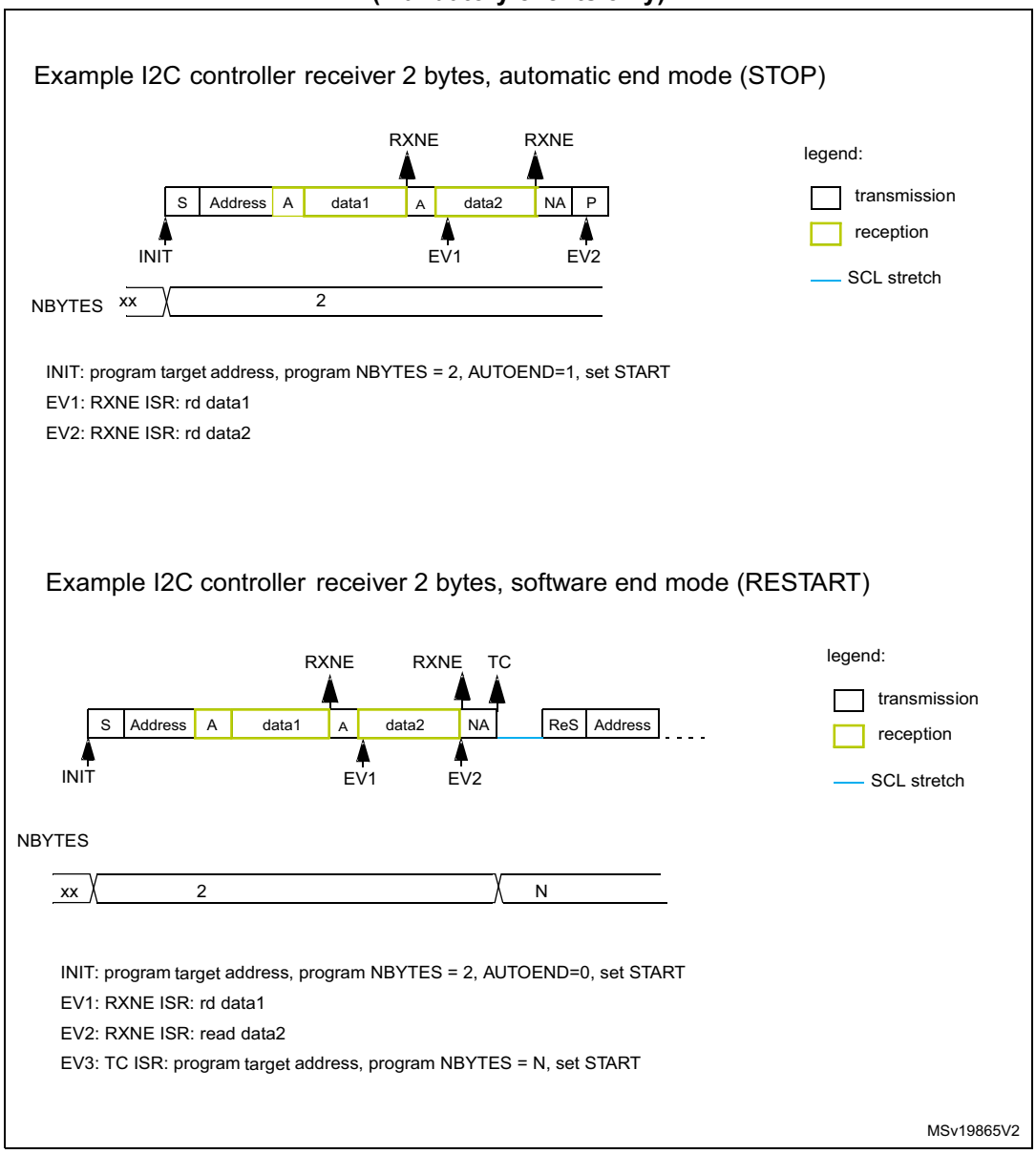


Figure 760. Transfer bus diagrams for I2C controller receiver (mandatory events only)



65.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 654. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 655. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

65.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

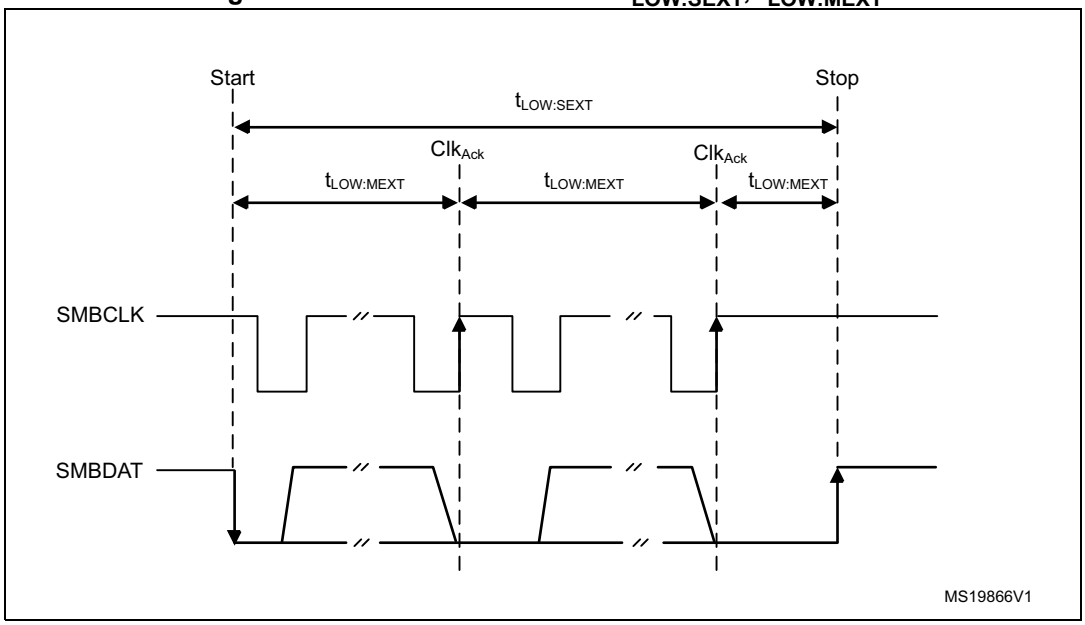
Table 656. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 761. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 65.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

65.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 657. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 658*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 659*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 660*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

65.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 658. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 659. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 660. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

65.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 65.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 762. Transfer sequence flow for SMBus target transmitter N bytes + PEC

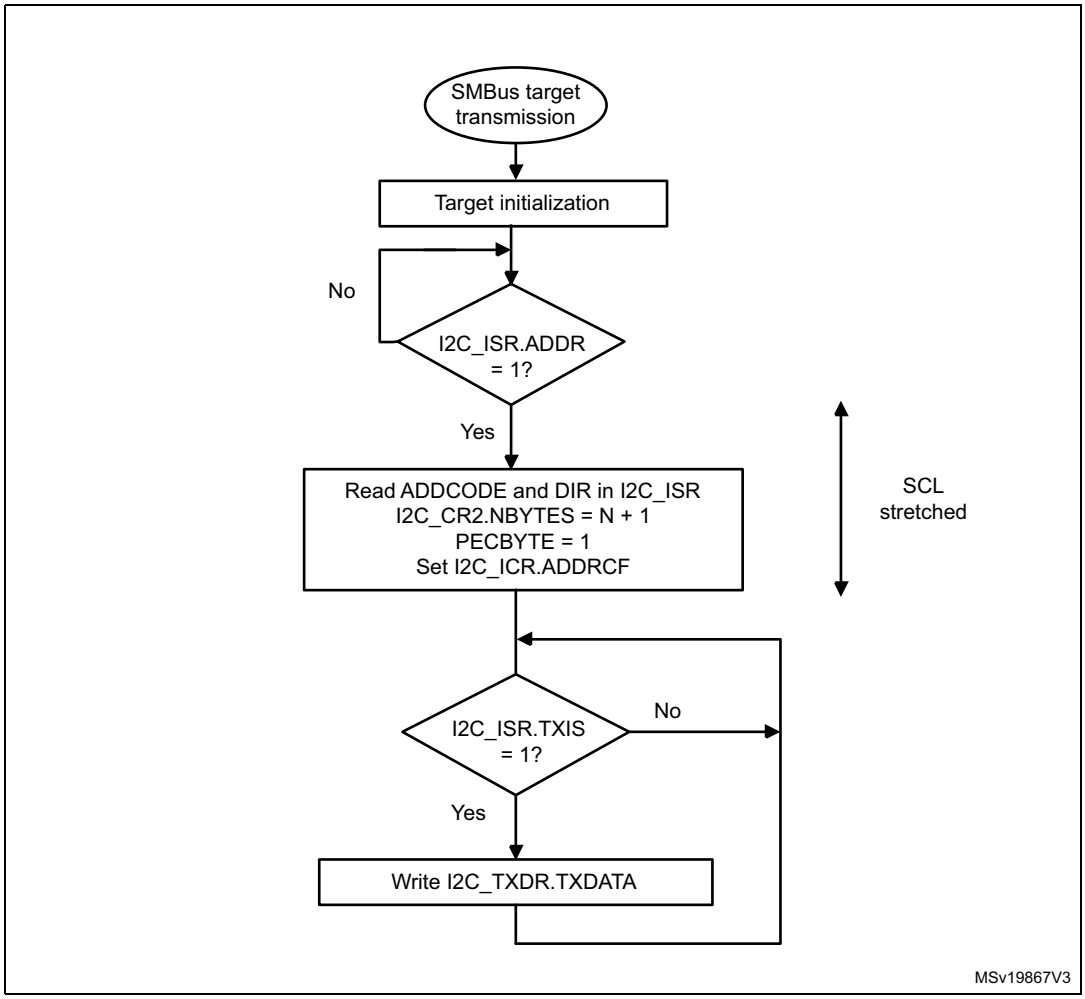
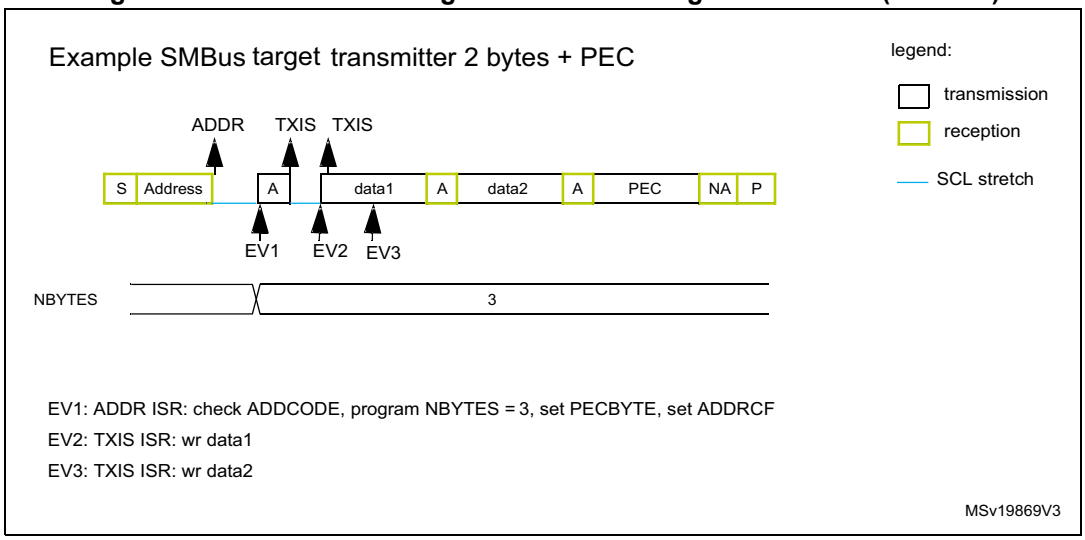


Figure 763. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 764. Transfer sequence flow for SMBus target receiver N bytes + PEC

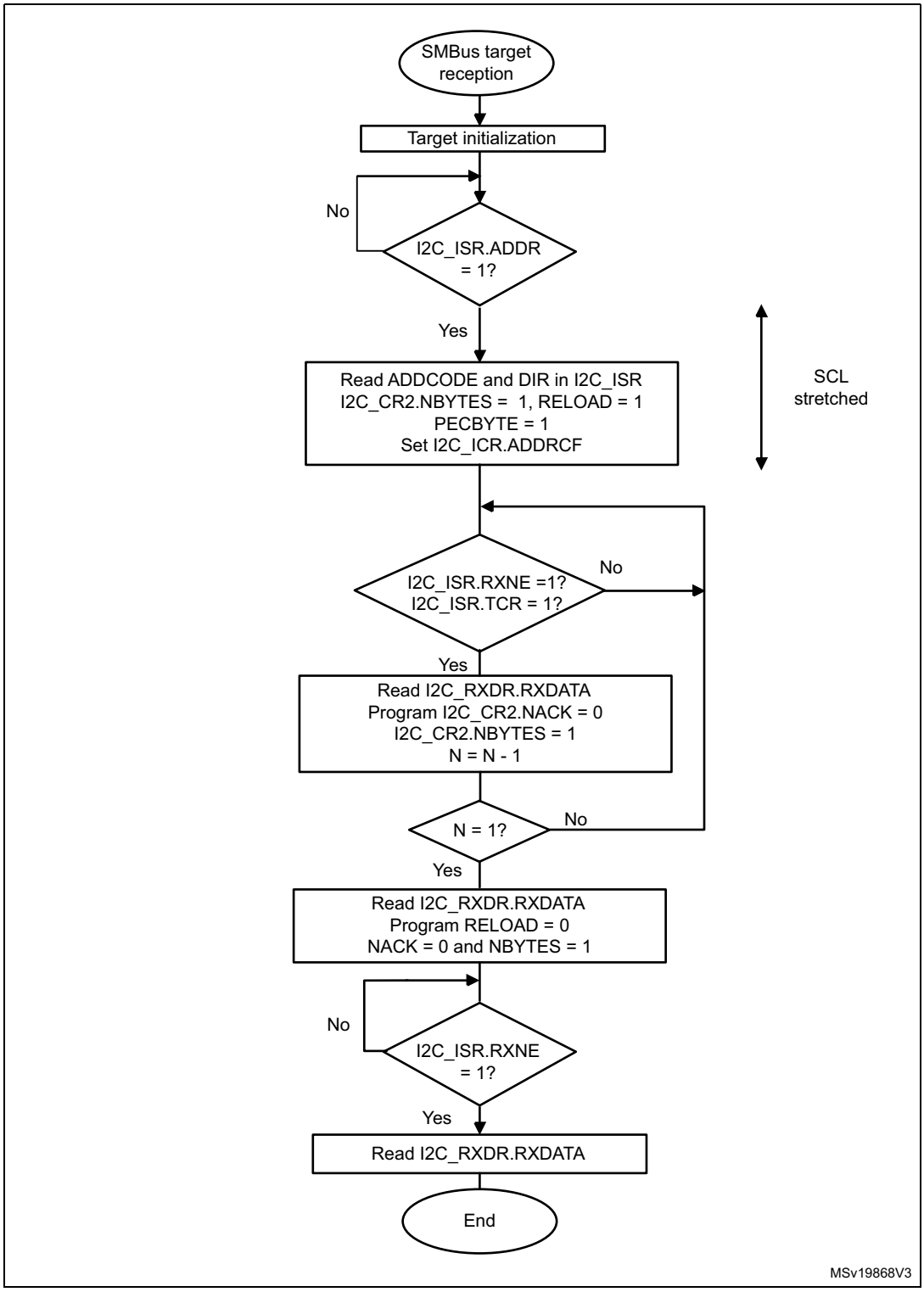
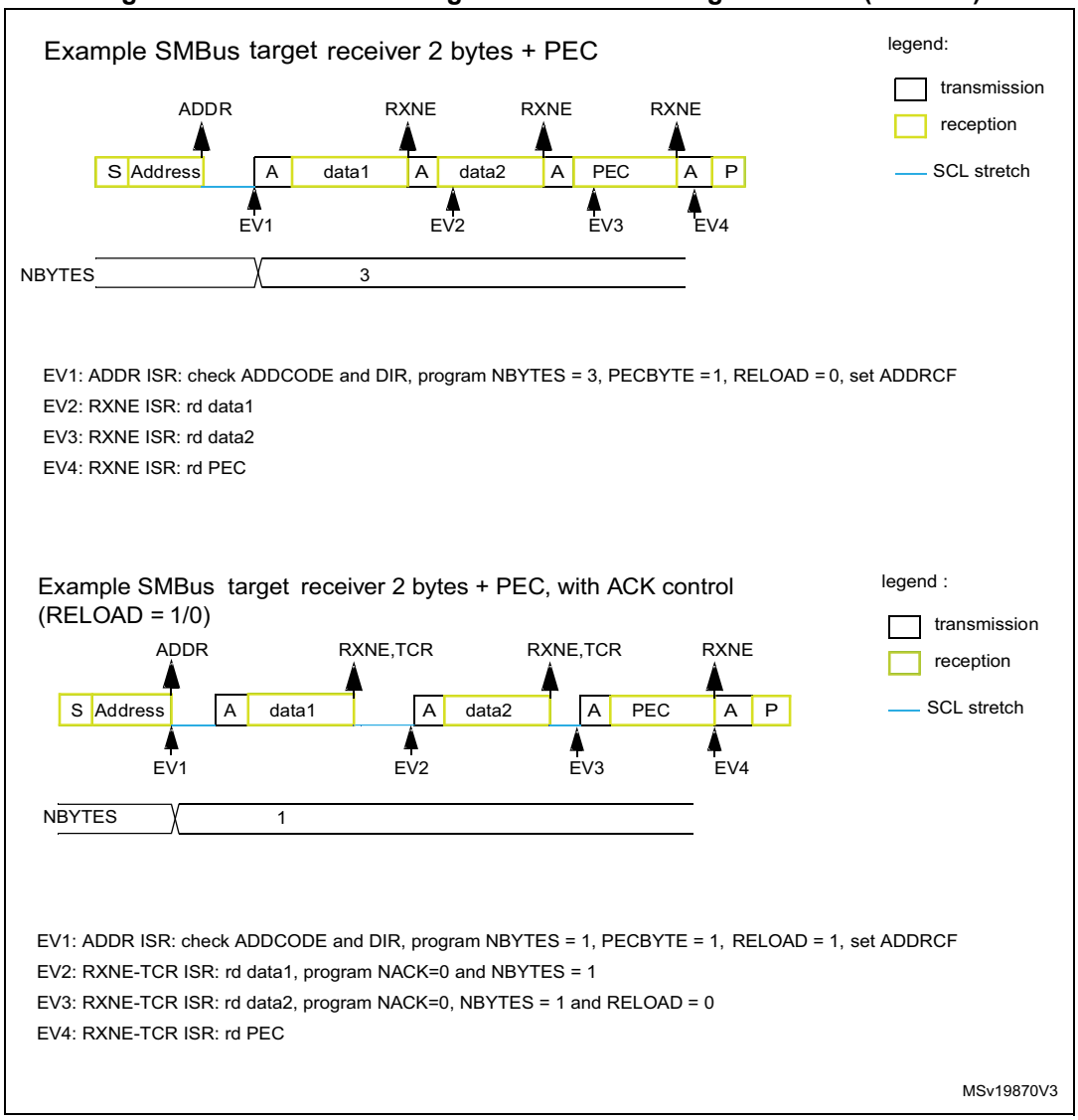


Figure 765. Bus transfer diagrams for SMBus target receiver (SBC = 1)



65.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 65.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

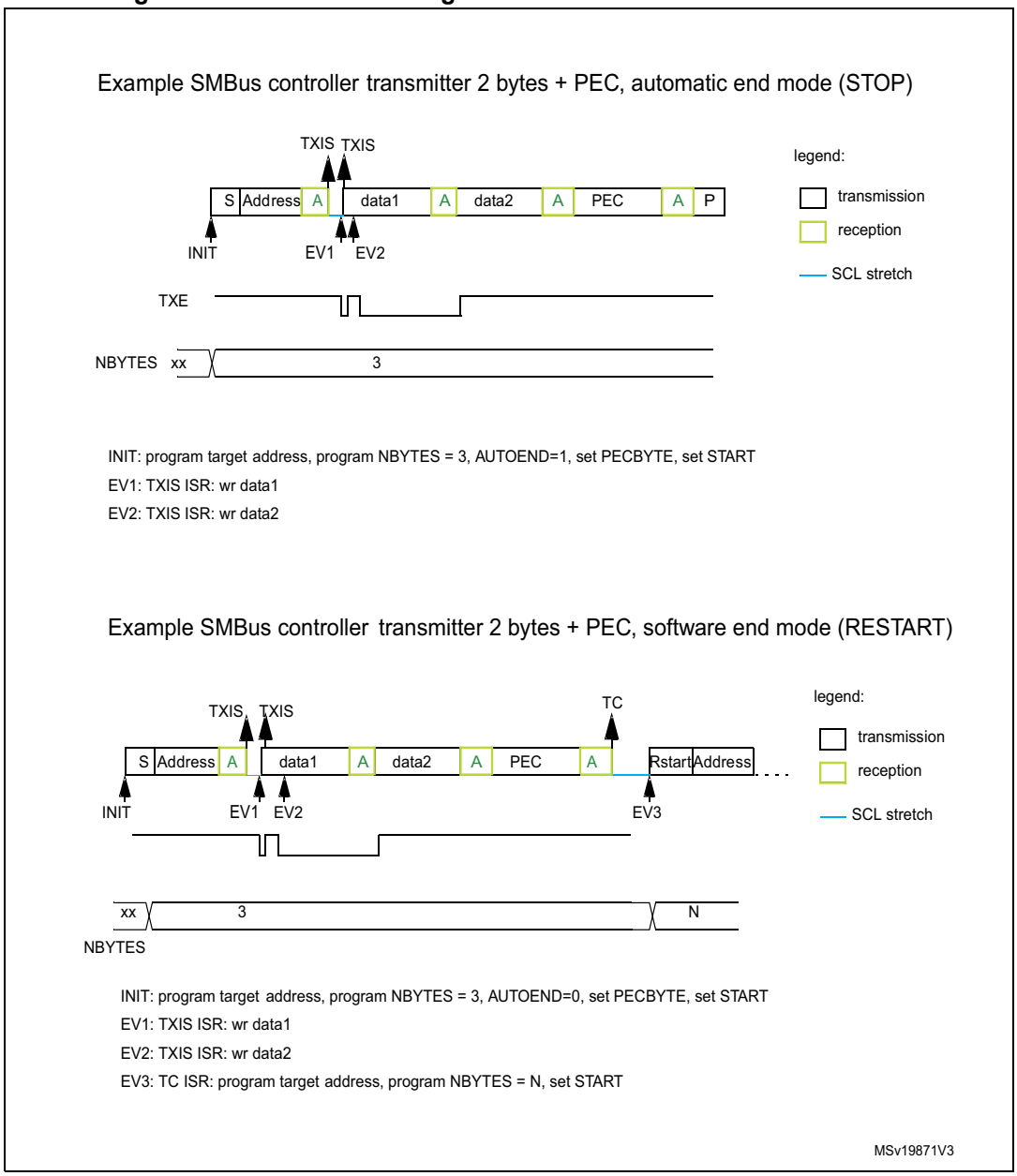
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 766. Bus transfer diagrams for SMBus controller transmitter



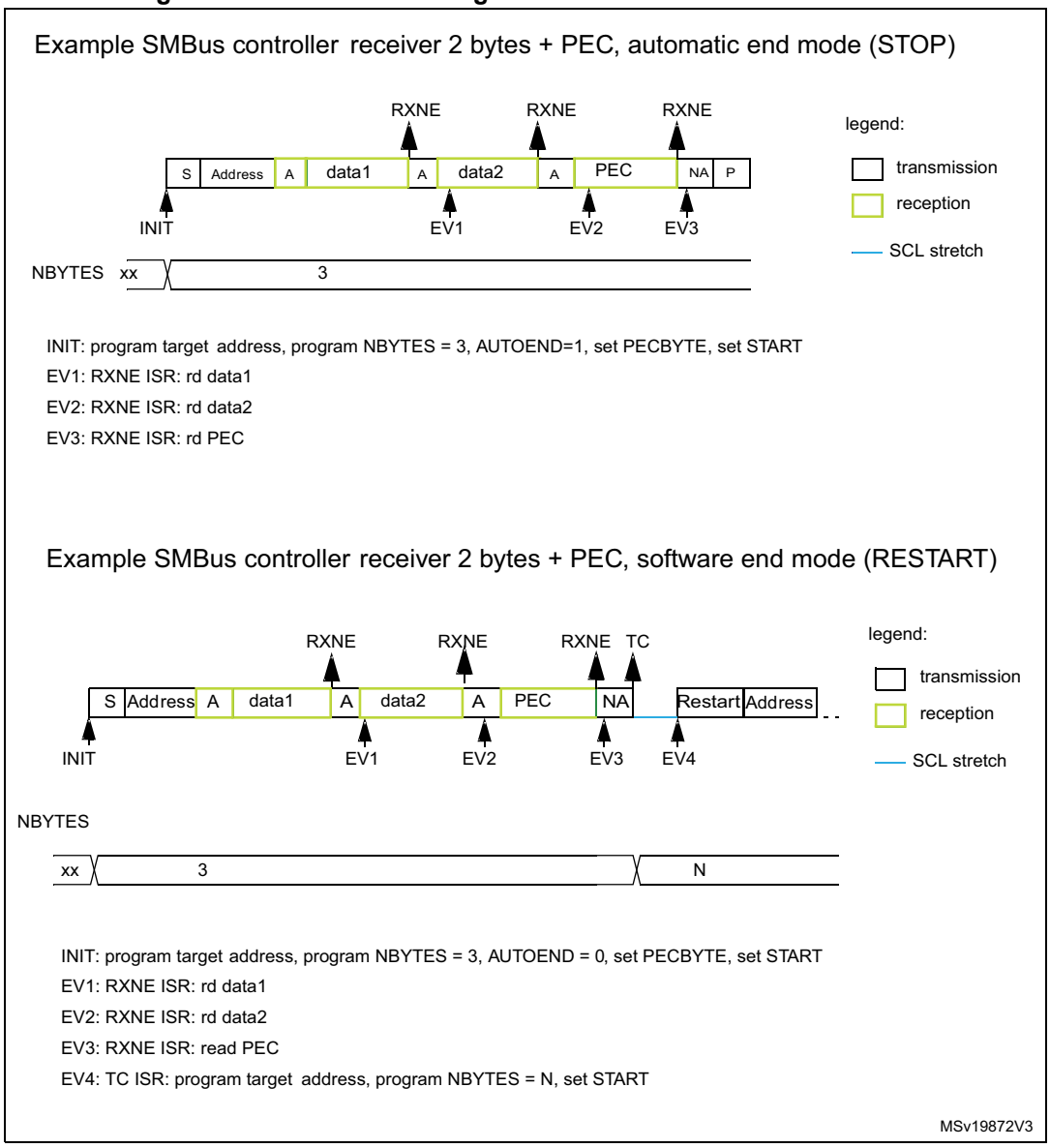
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 767. Bus transfer diagrams for SMBus controller receiver



65.4.16 Autonomous mode

The I2C peripheral can autonomously operate as controller or target in Stop mode as long as NOSTRETCH = 0. The autonomous mode is not supported when NOSTRETCH = 1. Besides Stop mode, the autonomous mode can also be used in Run and Sleep modes.

The APB clock is requested by the peripheral each time the I2C status needs to be updated. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the I2C configuration.

If an interrupt is generated, the device wakes up from Stop mode.

If there is no interrupt, the device remains in Stop mode, but the kernel and AHB/APB clocks are available for the I2C peripheral and all the autonomous peripherals enabled in the RCC.

If the DMA requests are enabled, the data are DMA-transferred to or from the SRAM, while the device remains in Stop mode.

Target mode

In target mode, the autonomous mode is enabled in Stop mode if WUPEN = 1 in the I2C_CR1. This mode is supported only when NOSTRETCH = 0.

The kernel clock is requested by the peripheral on START detection during all the transfer until the STOP condition occurs, when the target is addressed. If the target is not addressed, the kernel clock request is released after the address phase.

To optimize the functionality in target mode by using only DMA transfers, it is possible to set the ADDRACLR and STOPFACLR bits in the I2C_CR1 in order to avoid to serve the Address match (ADDR) and STOP detection (STOPF) events.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the internal oscillator is not switched on after a START is received in target mode.

Controller mode

In controller mode, a transfer can be automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger, selected by TRIGSEL in the I2C_AUTOCR register, generates a kernel clock request to allow the transfer, and launches a START condition and the I2C transfer as defined in the I2C_CR2 register. The kernel clock is requested until the STOP condition occurs.

To avoid waking up the CPU too often, it is possible to replace some interrupts by DMA requests, to make some control register write actions. In controller mode, *transfer complete* (TC) and *transfer complete reload* (TCR) events can generate a DMA request when the corresponding TCDMAEN or TCRDMAEN bits of the I2C_AUTOCR register are set. Consequently, the I2C_CR2 register can be written thanks to a DMA transfer, to program a new transfer (in TC event) or to reload the number of bytes (in TCR event).

In case a trigger is enabled, but the application needs to take the control back to start a different transfer, the following steps are required before writing in the I2C_CR2 register to launch the new transfer:

1. Wait for BUSY = 0 in the I2C_ISR register
2. Clear the TRIGEN bit of the I2C_AUTOCR register
3. Wait for longer than t_{BUF} (bus free time between a STOP and a START condition)
4. Wait for BUSY = 0 in the I2C_ISR register

Caution: When the device is in Stop mode, the I2C peripheral receives its kernel clock only when it is implicated in the transfer. Consequently, features such as timeouts and bus idle detection are not reliable in Stop mode.

Caution: The digital filter is not compatible with the functionality in Stop mode. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

65.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

65.5 I2C in low-power modes

Table 661. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. If the autonomous mode is enabled and I2C is clocked by an internal oscillator available in Stop mode: transfers in controller and in target modes are functional. DMA requests are functional, and the interrupts cause the device to exit the Stop mode. If WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby(2)	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 65.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific Stop mode is not supported, the instance must be disabled before entering that specific Stop mode.
2. Refer to *Section 65.3: I2C implementation* for information about LP-Stop, LPLV-Stop, and Standby modes supported by each instance.

65.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 662. I2C interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the LP-Stop, LPLV-Stop, Standby mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	Yes(1)	No(2)
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] = 0			
		Transfer complete	TC		Write START = 1 or STOP = 1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
	I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	Yes(1)	No(2)
		Arbitration loss	ARLO		Write ARLOCF=1			
		Overrun/underrun	OVR		Write OVRCF=1			
		PEC error	PECERR		Write PECERRCF=1			
		Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1			
		SMBus alert	ALERT		Write ALERTCF=1			

1. Refer to the I2C implementation table for information about wake-up from Stop mode support per instance.
2. Refer to I2C implementation table for information about wake-up from LP-Stop, LPLV-Stop, and Standby mode support per instance.

65.7 I2C DMA requests

65.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 25: High-performance direct memory access controller (HPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

65.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 25: High-performance direct memory access controller (HPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

65.7.3 Controller event control using DMA

In controller mode, the transfer can be automatically managed while the device is in Run, Sleep, or Stop mode, thanks to TC and TCR DMA requests.

If the TCDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when the TC bit of the I2C_ISR register is set. The DMA

controller must be programmed to write the next command in the I2C_CR2 register. If both STOP and START bits are set in the new I2C_CR2 command, a STOP condition followed by a START condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2. If only START bit is set in the new I2C_CR2 command, a RESTART condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2.

If the TCRDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when TCR bit of the I2C_ISR register is set. The DMA controller must be programmed to write the remaining number of bytes to transfer in the I2C_CR2 register.

65.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

65.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

65.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus drive enable

0: FMP I/O drive disabled

1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).

1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.

1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable

0: Host address disabled. Address 0b0001000x is NACKed.

1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.

1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.

1: Wake-up from Stop mode enabled.

Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.

0: Target byte control disabled

1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled

Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

65.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.
0: No PEC transfer
1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.
This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)
This bit is set and cleared by software.
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
This bit is set and cleared by software.
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)
0: Controller requests a write transfer
1: Controller requests a read transfer
Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)
Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.
Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

65.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable
0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.
Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address
7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.
Note: These bits can be written only when OA1EN = 0.

65.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable

0: Own address 2 disabled. The received target address OA2 is NACKed.

1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

000: No mask

001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.

010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.

011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.

100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.

101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.

110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.

111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

65.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

65.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

- Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
- Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

65.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

65.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear

Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear

Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear

Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear

Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear

Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear

Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

65.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

65.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

65.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

65.9.12 I2C autonomous mode control register (I2C_AUTOCR)

Address offset: 0x2C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Trigger enable

0: Trigger disabled

1: Trigger enabled

When a trigger is detected, a START condition is sent and the transfer is launched as defined in I2C_CR2.

Bit 20 **TRIGPOL:** Trigger polarity

0: Trigger active on rising edge

1: Trigger active on falling edge

Note: This bit can be written only when PE = 0

Bits 19:16 **TRIGSEL[3:0]:** Trigger selection (refer to *Section 65.4.2: I2C pins and internal signals* I2C interconnections tables).
0000: i2c_trg0 selected
0001: i2c_trg1 selected
...
1111: i2c_trg15 selected
Note: This bit can be written only when PE = 0

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TCRDMAEN**: DMA request enable on Transfer Complete Reload event
0: DMA request not generated on Transfer Complete Reload event
1: DMA request generated on Transfer Complete Reload event

Bit 6 **TCDMAEN**: DMA request enable on Transfer Complete event
0: DMA request not generated on Transfer Complete event
1: DMA request generated on Transfer Complete event

Bits 5:0 Reserved, must be kept at reset value.

65.9.13 I2C hardware configuration register (I2C_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTON[3:0]				WKP[3:0]				ASYN[3:0]				SMBUS[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **AUTON[3:0]**:
0: Autonomous mode not implemented
1: Autonomous mode implemented

Bits 11:8 **WKP[3:0]**:
0: Wake-up from Stop mode not implemented
1: Wake-up from Stop mode implemented

Bits 7:4 **ASYN[3:0]**:
0: Independent kernel clock not implemented
1: Independent kernel clock implemented

Bits 3:0 **SMBUS[3:0]**:
0: SMBus mode not implemented
1: SMBus mode implemented

65.9.14 I2C version register (I2C_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

65.9.15 I2C identification register (I2C_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier.

65.9.16 I2C size identification register (I2C_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier.

65.9.17 I2C register map

The table below provides the I2C register map and the reset values.

Table 663. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 663. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	I2C_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0									0	0						
0x030-0x03EC	Reserved																																
0x03F0	I2C_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AUTON[3:0]				WKP[3:0]				ASYN[3:0]				SMBUS[3:0]			
	Reset value																	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0x03F4	I2C_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	1
0x03F8	I2C_IPIDR	ID[31:16]																ID[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0x03FC	I2C_SIDR	SID[31:16]																SID[15:0]															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

66 Improved inter-integrated circuit (I3C)

66.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 669: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus.

The I3C peripheral can be used with DMA, to off-load the CPU.

66.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 669*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Sm/Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in *Table 675*)
  - I3C SDR read/write direct CCC messages (see details in *Table 675*)

- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 66.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 66.3.2*):
  - On a reset pattern
  - On a missed start
- Autonomous mode and transfers during Stop mode(s) with DMA (see *Section 66.3.3*)
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 66.6.2*)

66.3 I3C implementation

66.3.1 I3C instantiation

There are several I3C instances in the device, all with the same feature set:

Table 664. I3C instantiation

Instance(s)
I3C1, I3C2, I3C3

66.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 665*. For more details about the wake-up capabilities, refer to *Section 66.13*.

Table 665. I3C wake-up

Wake-up
From Stop1/2 and LP-Stop1/2

66.3.3 I3C autonomous mode with DMA in Stop mode(s)

The I3C peripheral can be assisted with an autonomous DMA, functional in Stop mode, so that I3C transfer(s) is (are) performed over the I3C bus, as written to and/or read from the I3C FIFO-based registers via the programmed DMA channels, without involving the CPU. Afterwards, the I3C or/and DMA can be used to wake up the device from Stop mode.

Table 666 indicates the I3C instance(s) and DMA(s) able to operate autonomously in a Stop mode.

Table 666. I3C autonomous mode with DMA in Stop mode(s)

Instance	Autonomous mode with DMA
I3C4	In Stop1/2, LP-Stop1/2, and LPLV-Stop1/2 modes (with LPDMA1)
I3C1, I3C2, I3C3	In Stop1/2 modes (with HPDMA1/2/3)

66.3.4 I3C FIFOs

The FIFOs are implemented as defined in *Table 667*.

Table 667. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

66.3.5 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

66.3.6 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 668*.

Table 668. I3C interrupt(s)

Signal name	Signal type	Description
i3c_it	O	Global interrupt line

66.3.7 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 669*.

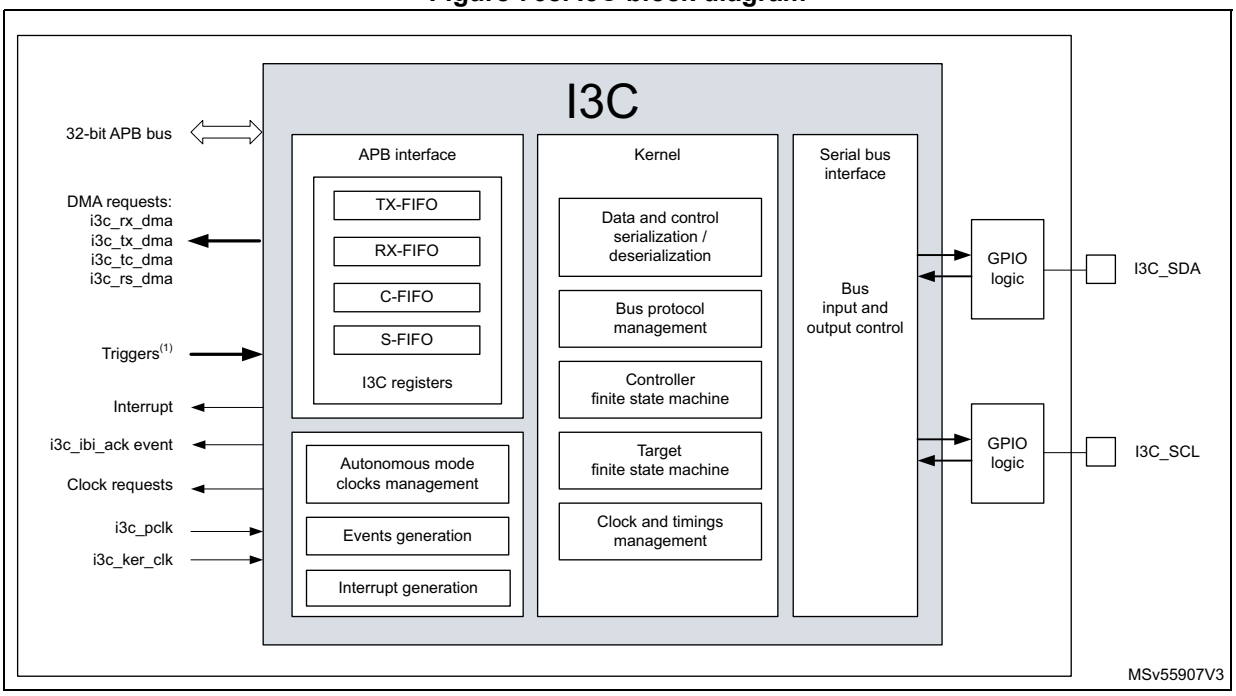
Table 669. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	-	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA when controller and target. Supported broadcast SETAASA and direct SETDASA when controller.
7-bit static I^2C address	X	X	-	No support of the I3C peripheral as target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 675* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller (refer to *Timing control and sensor acquisition timestamping when I3C controller*). Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

66.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 768*.

Figure 768. I3C block diagram



1. Refer to *Section 66.3.5: I3C triggers*.

66.5 I3C pins and internal signals

Table 670. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Table 671. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line

Table 671. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller
i3c_ibi_ack	O	IBI acknowledge event when the I3C peripheral acts as controller, for timing control with asynchronous mode 0

1. Refer to *Section 66.3.6: I3C interrupt(s)*.

66.6 I3C reset and clocks

66.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 675* and *Section 66.16.17* for more details.

This reset interrupt notification can be used to wake up from a low power mode. For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

66.6.2 I3C clocks and requirements

As indicated in *Figure 768*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

66.7 I3C peripheral state and programming

66.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 769* and *Figure 770*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

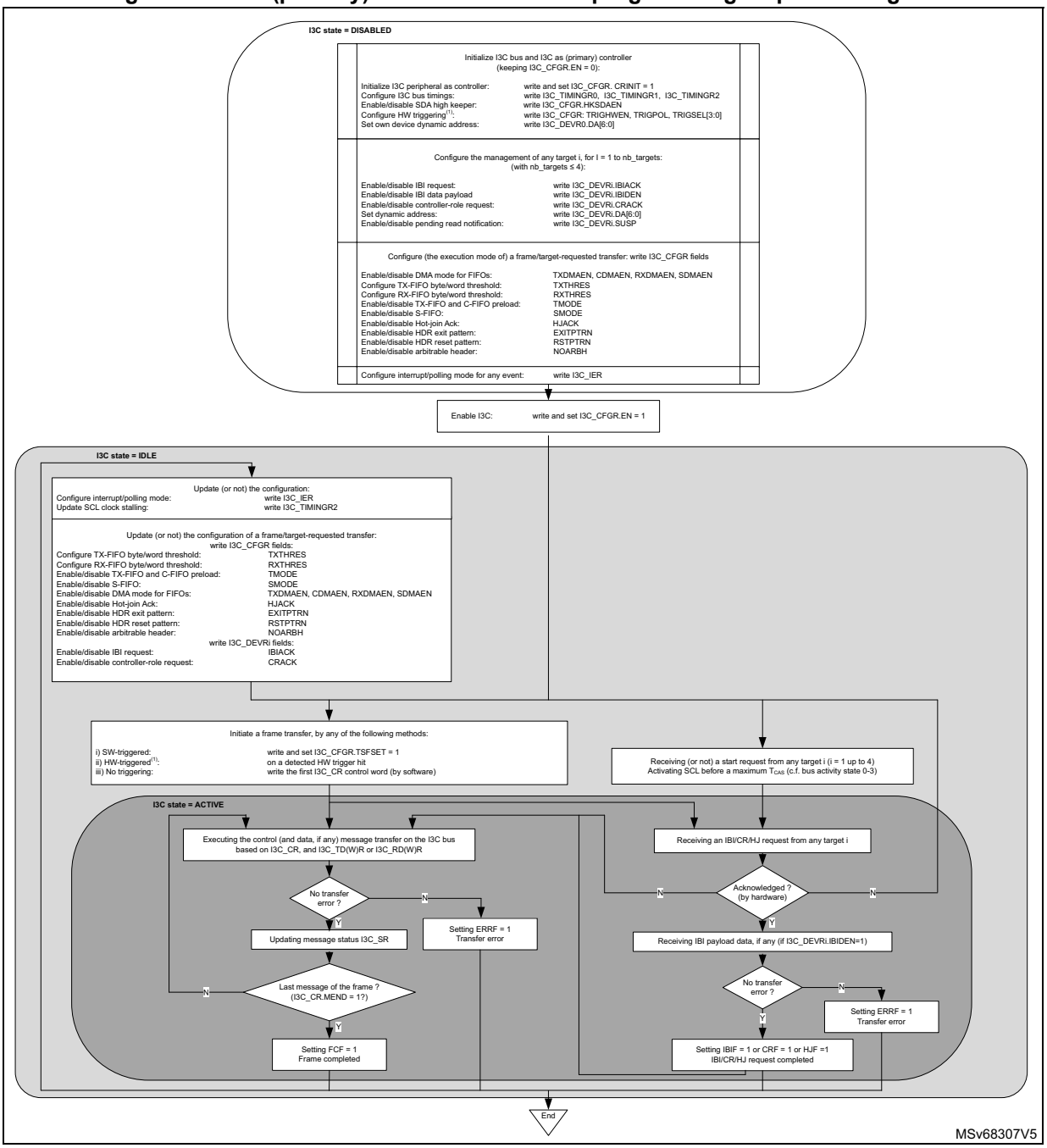
Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

Note: The software can disable the generation of the FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 66.3), to stream a series of frames.

66.7.2 I3C controller state and programming sequence

Figure 769 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 769. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 66.3.5: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
  - FCFDIS: FCF generation enable/disable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 66.9*:

- *Figure 781: IBI transfer, as controller/target*
- *Figure 782: Hot-join request transfer, as controller/target*
- *Figure 783: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 66.9*:

- *Figure 771: I3C CCC messages, as controller*
- *Figure 772: I3C broadcast ENTDAA CCC, as controller*
- *Figure 773: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 778: I3C private read/write messages, as controller*
- *Figure 780: Legacy I2C read/write messages, as controller*

Figure 769 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 66.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 66.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
  - FCFDIS

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

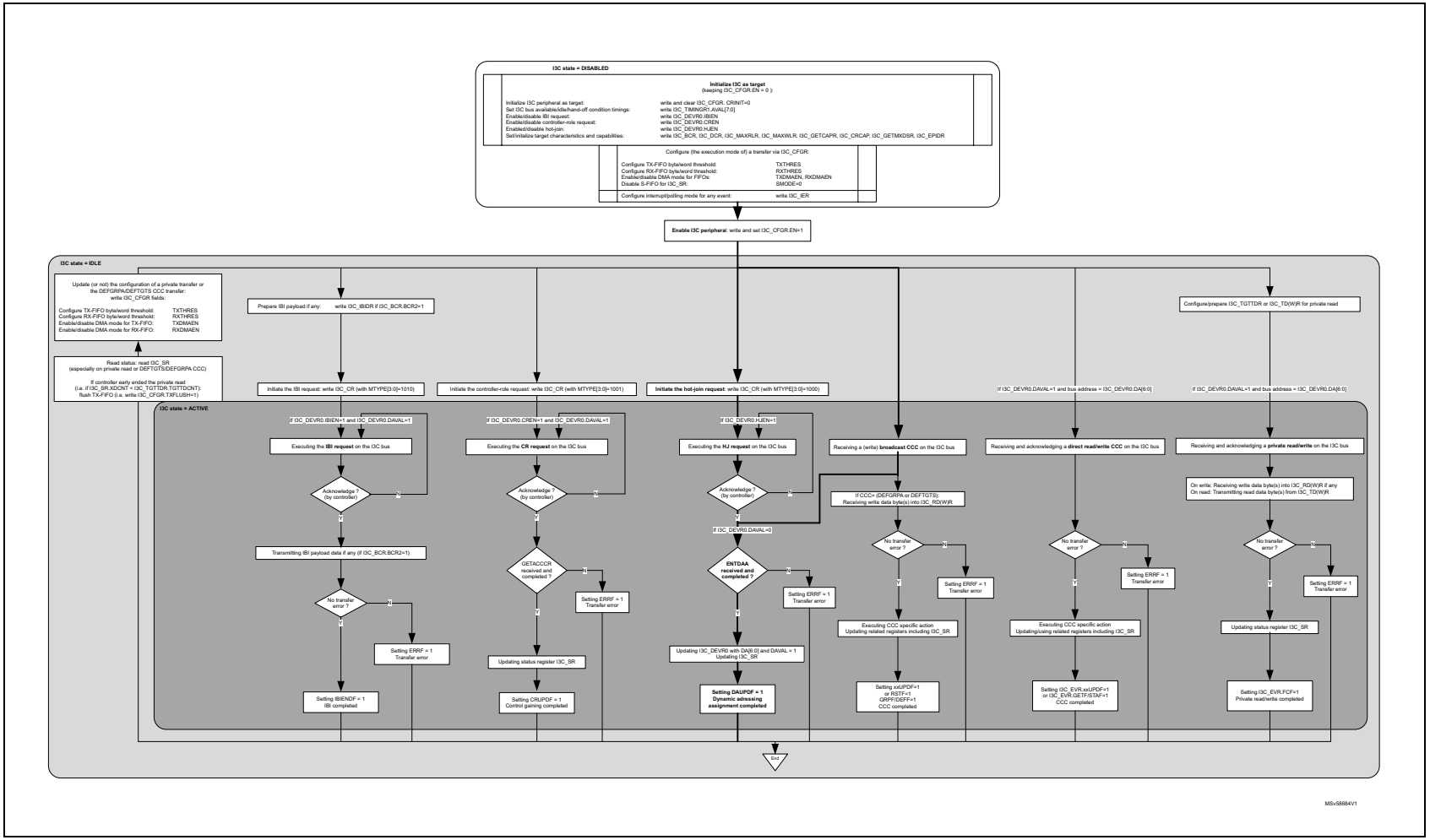
The registers usage versus the I3C peripheral role as controller is summarized in *Section 66.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 673*.

66.7.3 I3C target state and programming sequence

Figure 770 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 770. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 66.9*:

- *Figure 774: I3C CCC messages, as target*
- *Figure 775: I3C broadcast ENTDAA CCC, as target*
- *Figure 776: I3C broadcast DEFTGTS CCC, as target*
- *Figure 777: I3C broadcast DEFGRPA CCC, as target*
- *Figure 779: I3C private read/write messages, as target*

Figure 770 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 66.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 66.9*:

- *Figure 781: IBI transfer, as controller/target*
- *Figure 782: Hot-join request transfer, as controller/target*
- *Figure 783: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC)

The registers usage vs. the I3C peripheral role as target is summarized in *Section 66.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 674*.

66.8 I3C registers and programming

66.8.1 I3C register set, as controller/target

Table 672 lists the registers and their usage versus the I3C peripheral role.

Table 672. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_MISR	X	X

Table 672. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVR0	X	X
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

66.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 673* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 673. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 673. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) FCFDIS CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 673. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_MISR	X	-	-	Read (controller-role fields): HJMIS CRMIS IBIMIS FCMIS ERRMIS RXTGTENDMIS RXFNEMIS TXFNFMIS SFNEMIS CFNFMIS
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			

Table 673. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

66.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 674* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 674. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 674. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_MISR	X			Read (target-role fields): GRPMIS DEFMIS INTUPDMIS ASUPDMIS RSTMIS MRLUPDMIS MWLUPDMIS DAUPDMIS STAMIS GETMIS WKPMIS CRUPDMIS IBIENDMIS ERRMIS FCMIS RXFNEMIS TXFNFMIS

Table 674. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

66.9 I3C bus transfers and programming

66.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 675*.

Table 675. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 776*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	-	-
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 777*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 675. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 66.16.20*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 66.16.19*.

Table 675. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 66.16.29*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 66.16.24*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 66.16.25*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 66.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 66.16.28*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 66.16.26*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 66.16.27*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

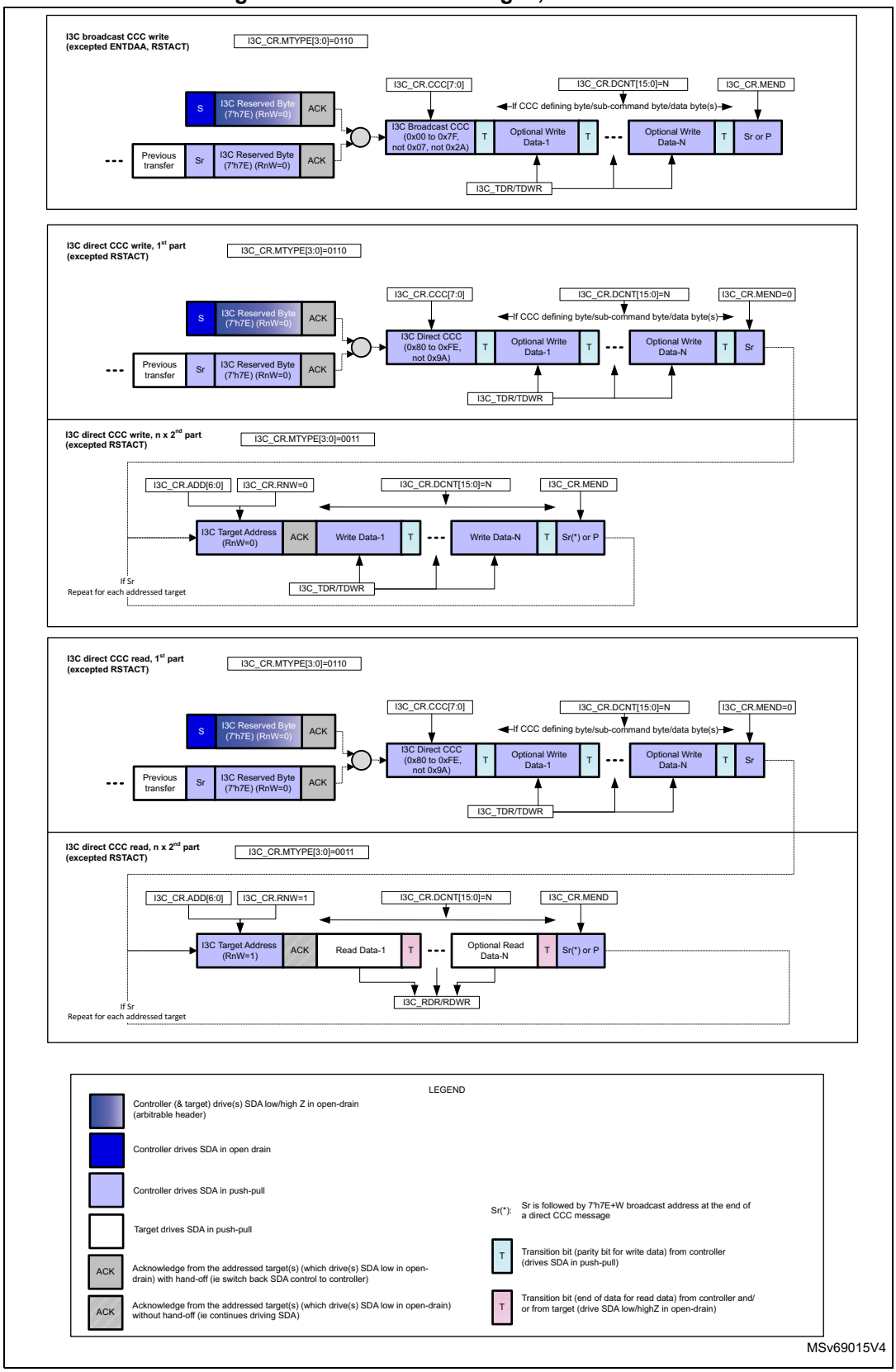
Table 675. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

66.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 771 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

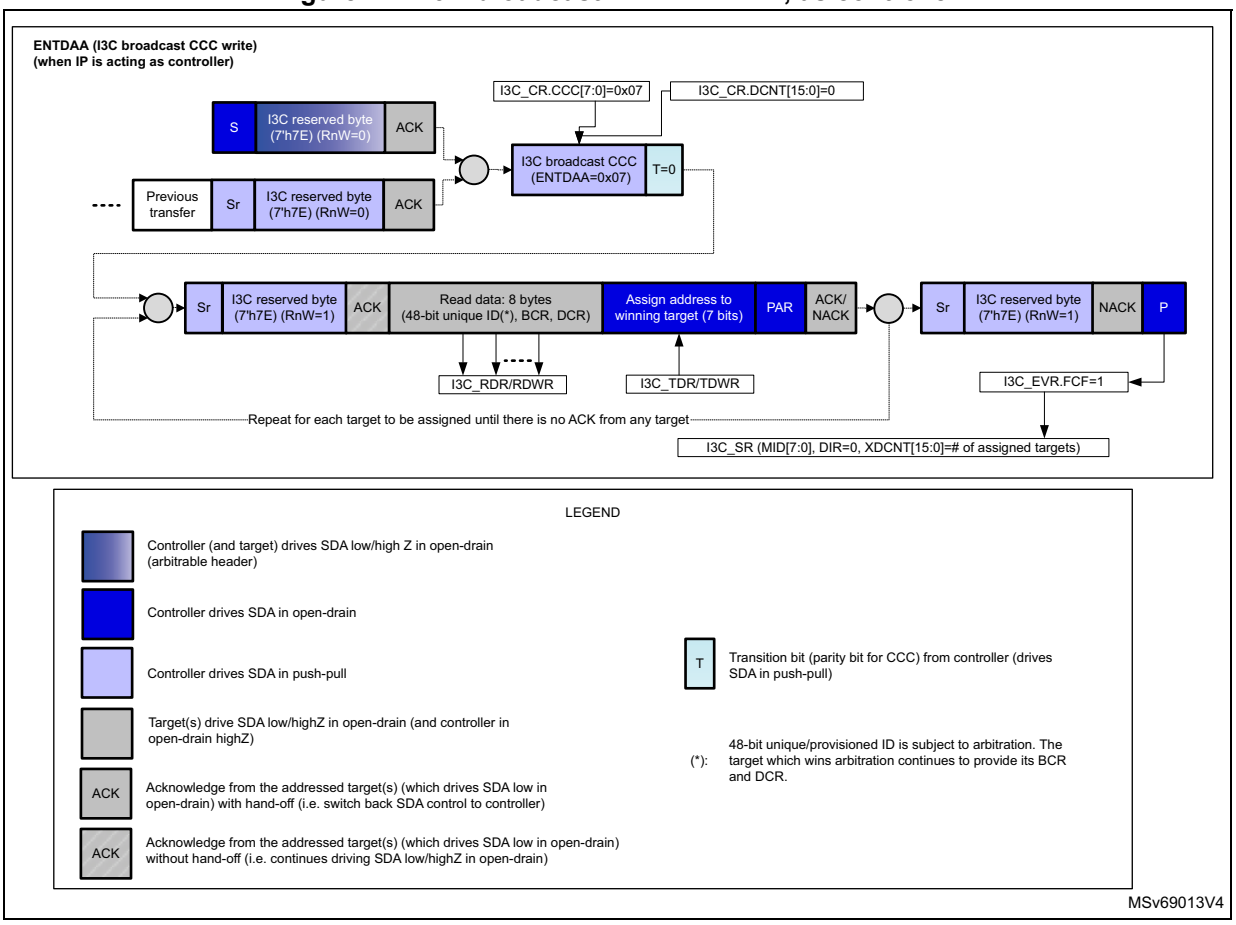
Figure 771. I3C CCC messages, as controller



66.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 772 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

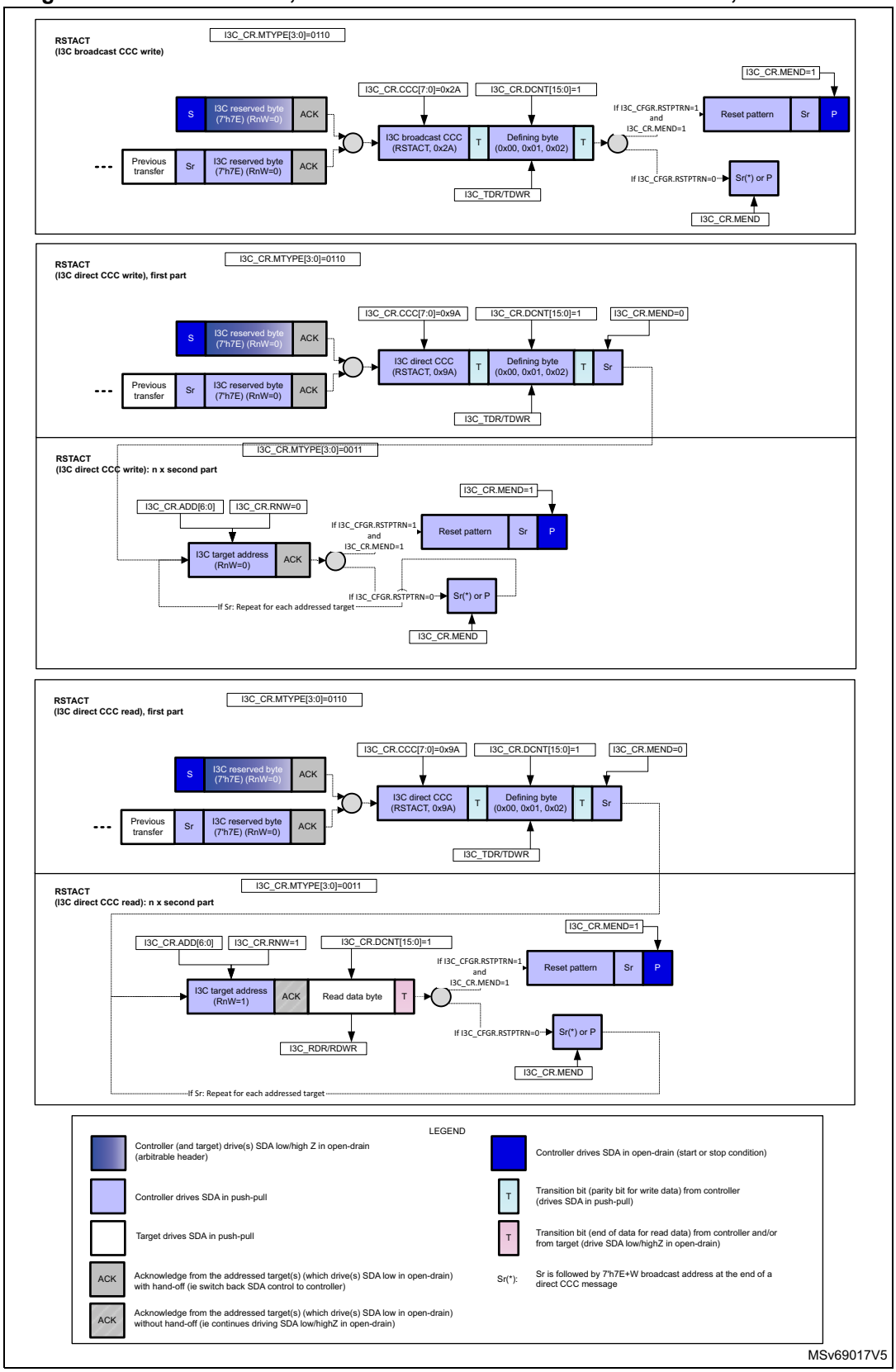
Figure 772. I3C broadcast ENTDAA CCC, as controller



66.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 773 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

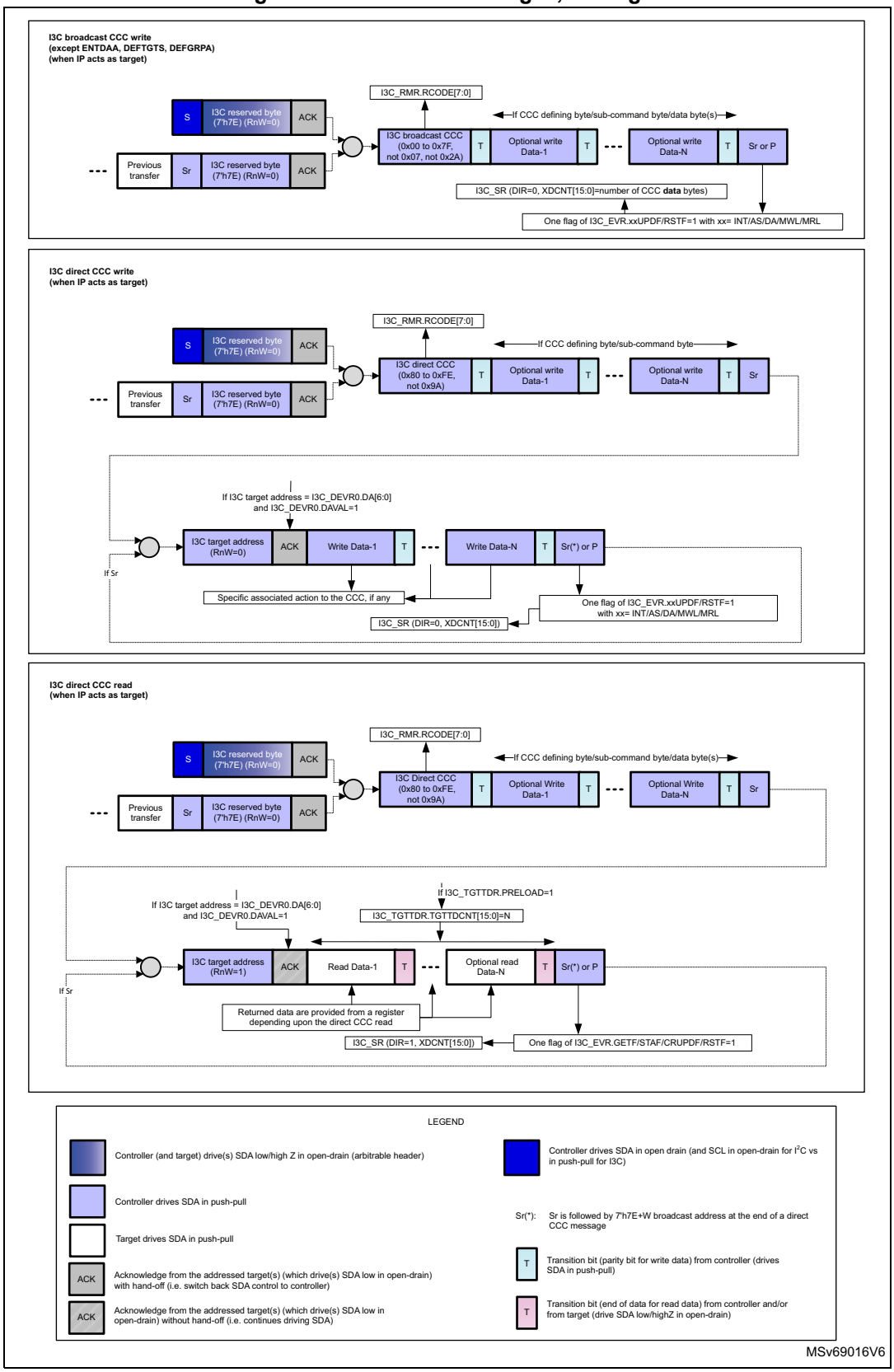
Figure 773. I3C broadcast, direct read and direct write RSTACT CCC, as controller



66.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 774 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

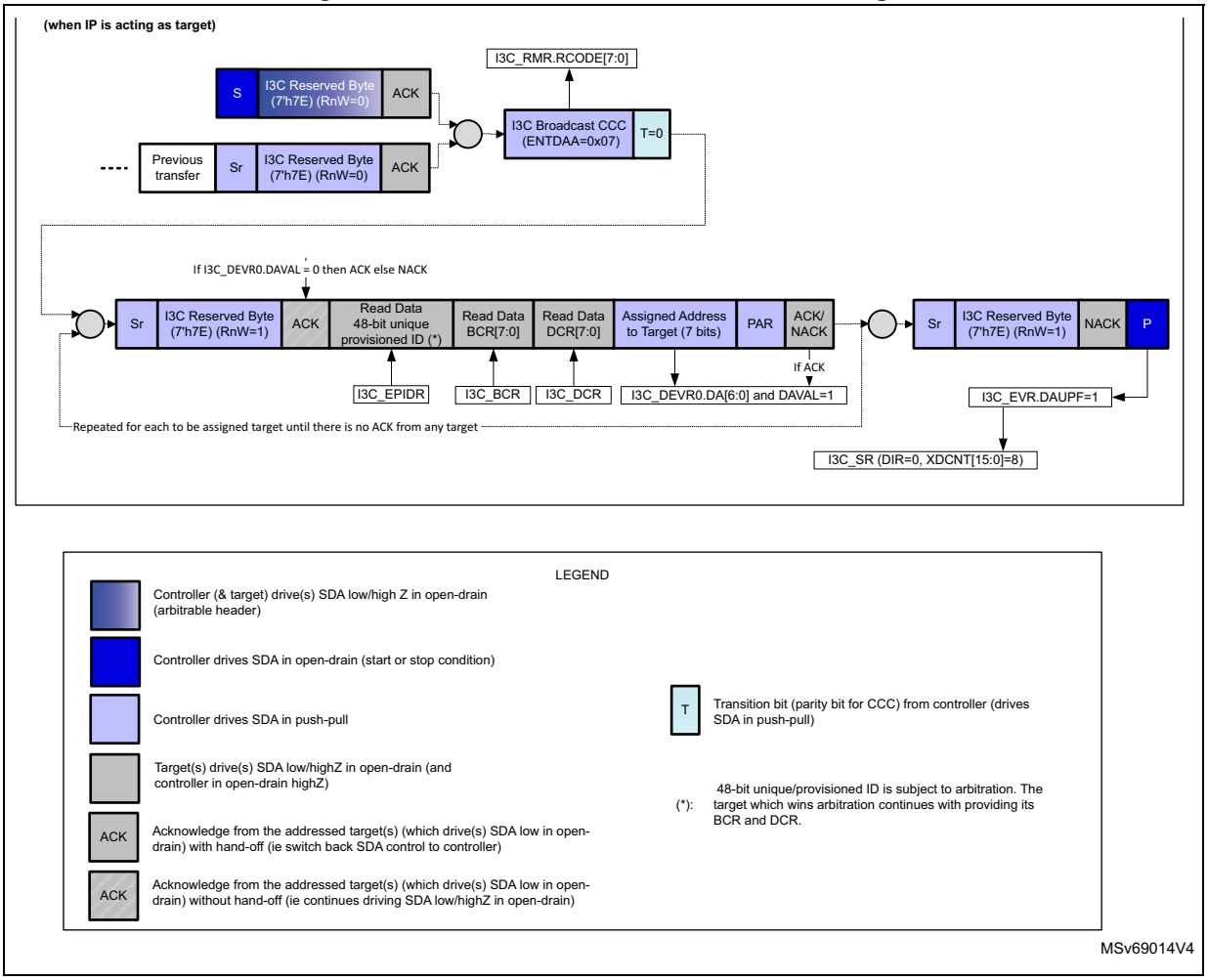
Figure 774. I3C CCC messages, as target



66.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 775 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

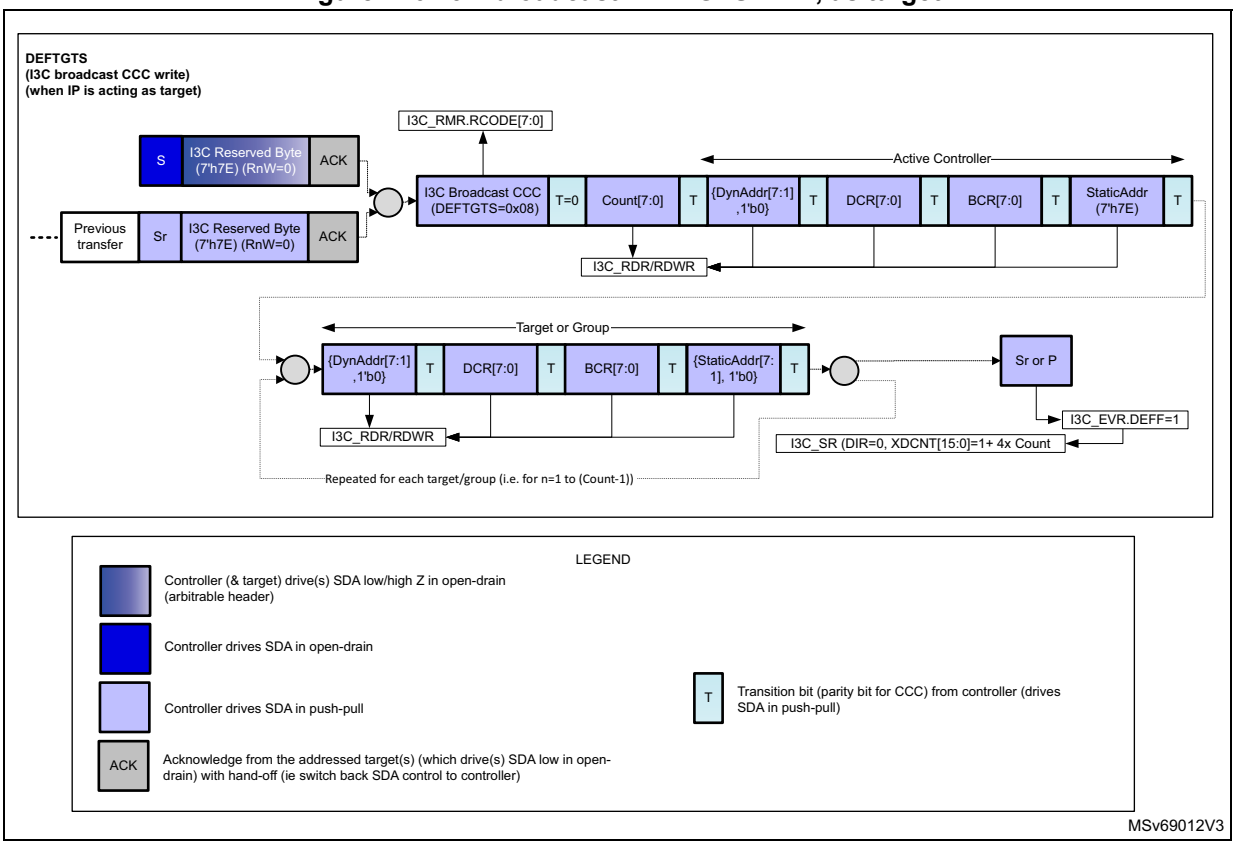
Figure 775. I3C broadcast ENTDAA CCC, as target



66.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 776 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

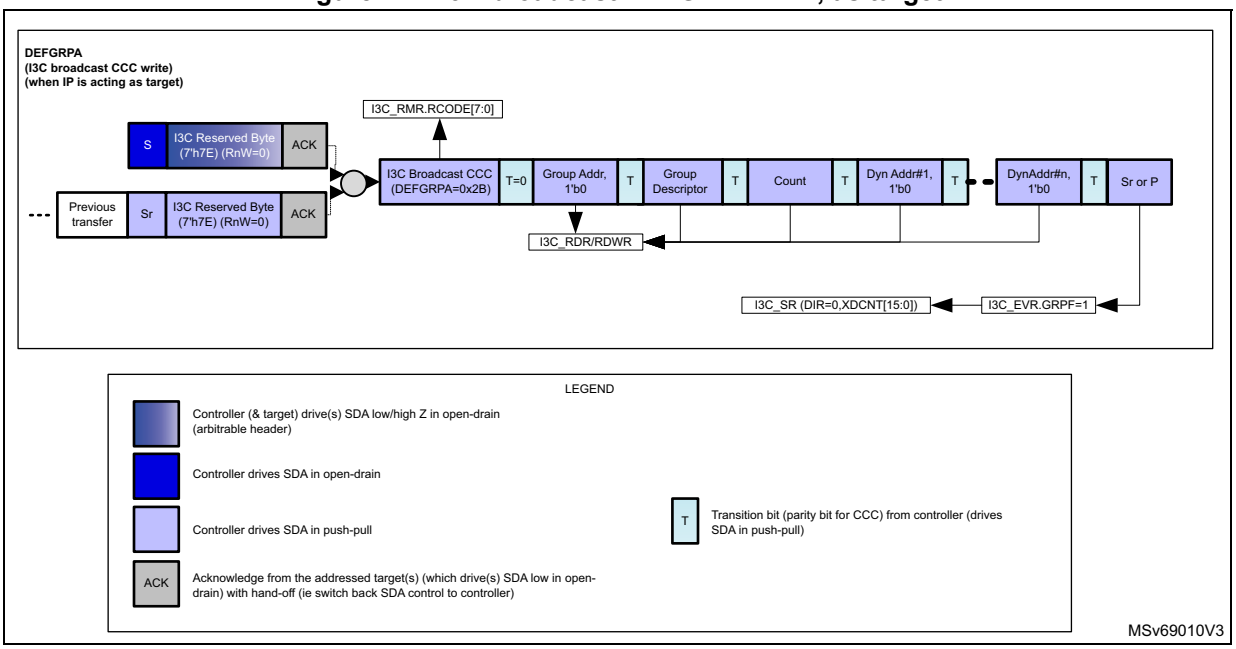
Figure 776. I3C broadcast DEFTGTS CCC, as target



66.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 777 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 777. I3C broadcast DEFGRPA CCC, as target



66.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

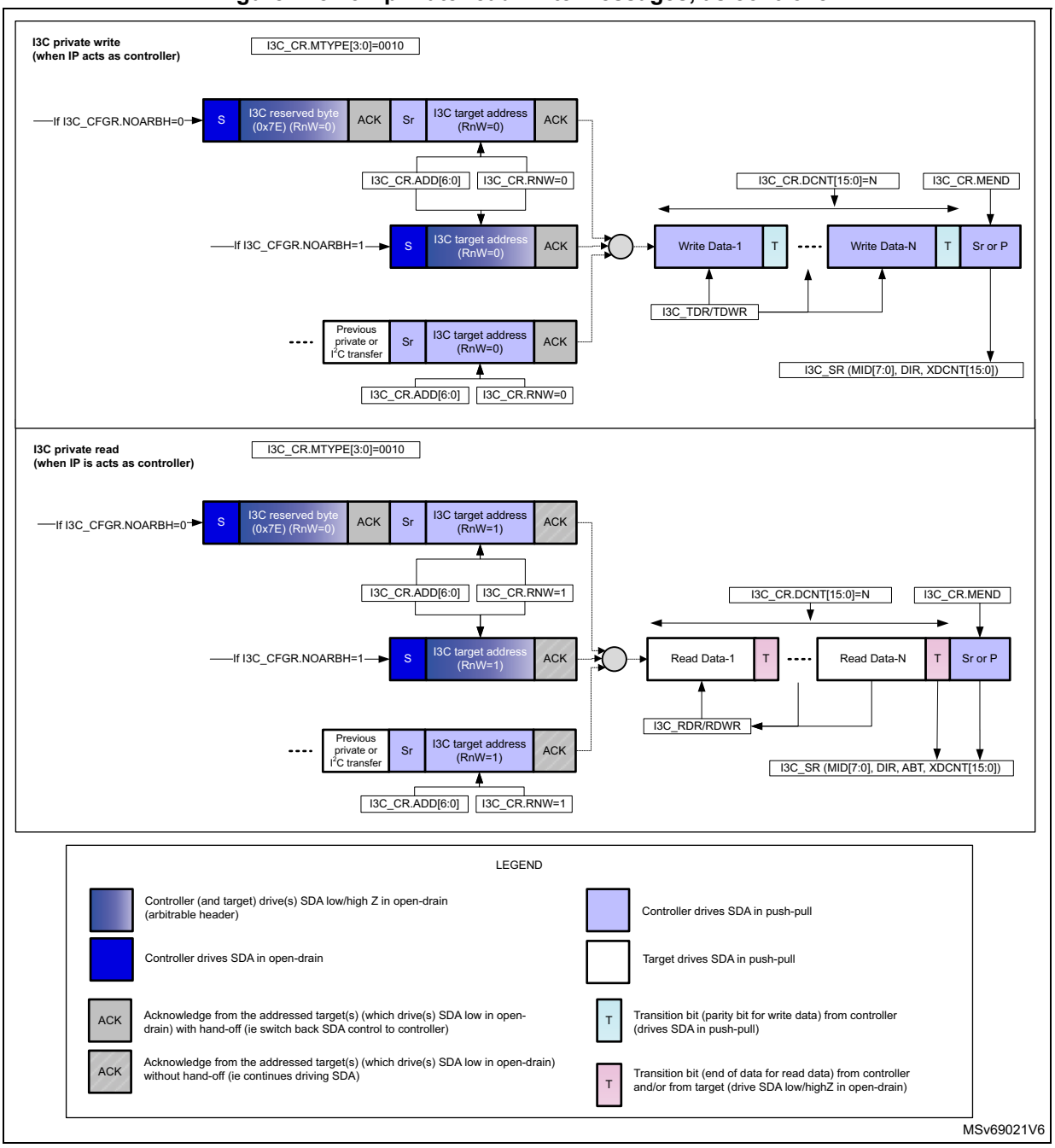
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

66.9.10 I3C private read/write transfer, as controller

Figure 778 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

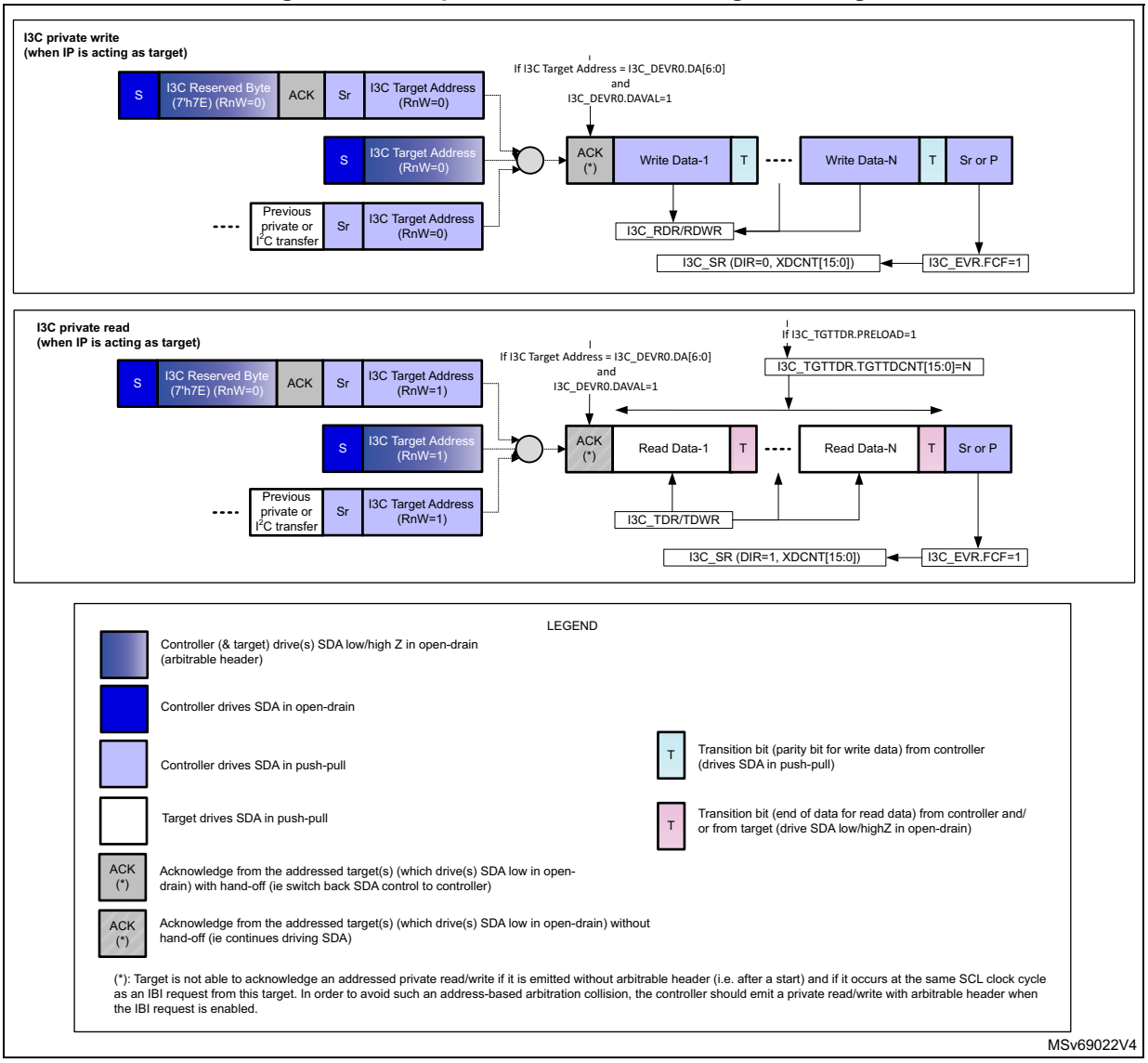
Figure 778. I3C private read/write messages, as controller



66.9.11 I3C private read/write transfer, as target

Figure 779 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

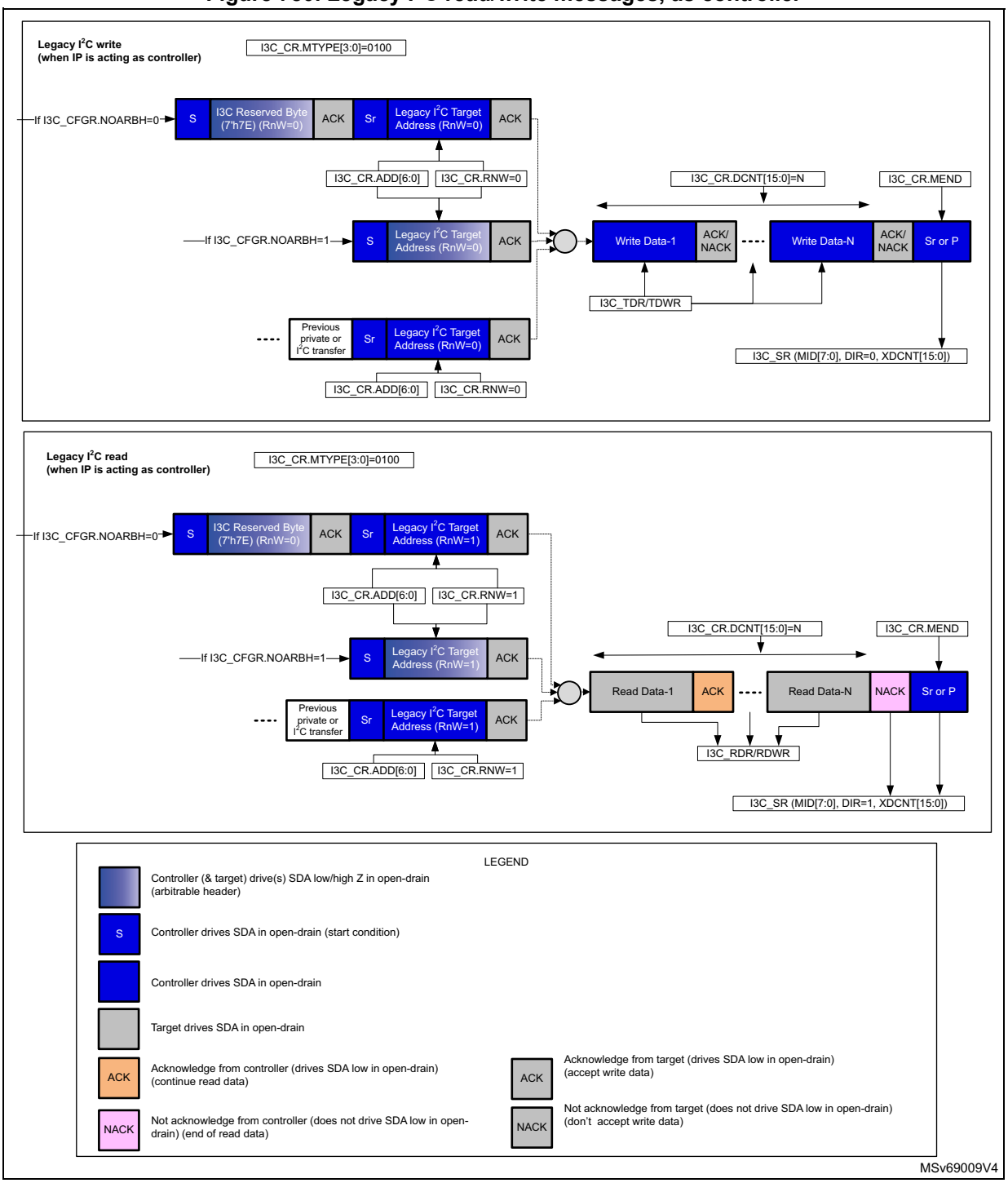
Figure 779. I3C private read/write messages, as target



66.9.12 Legacy I²C read/write transfer, as controller

Figure 780 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

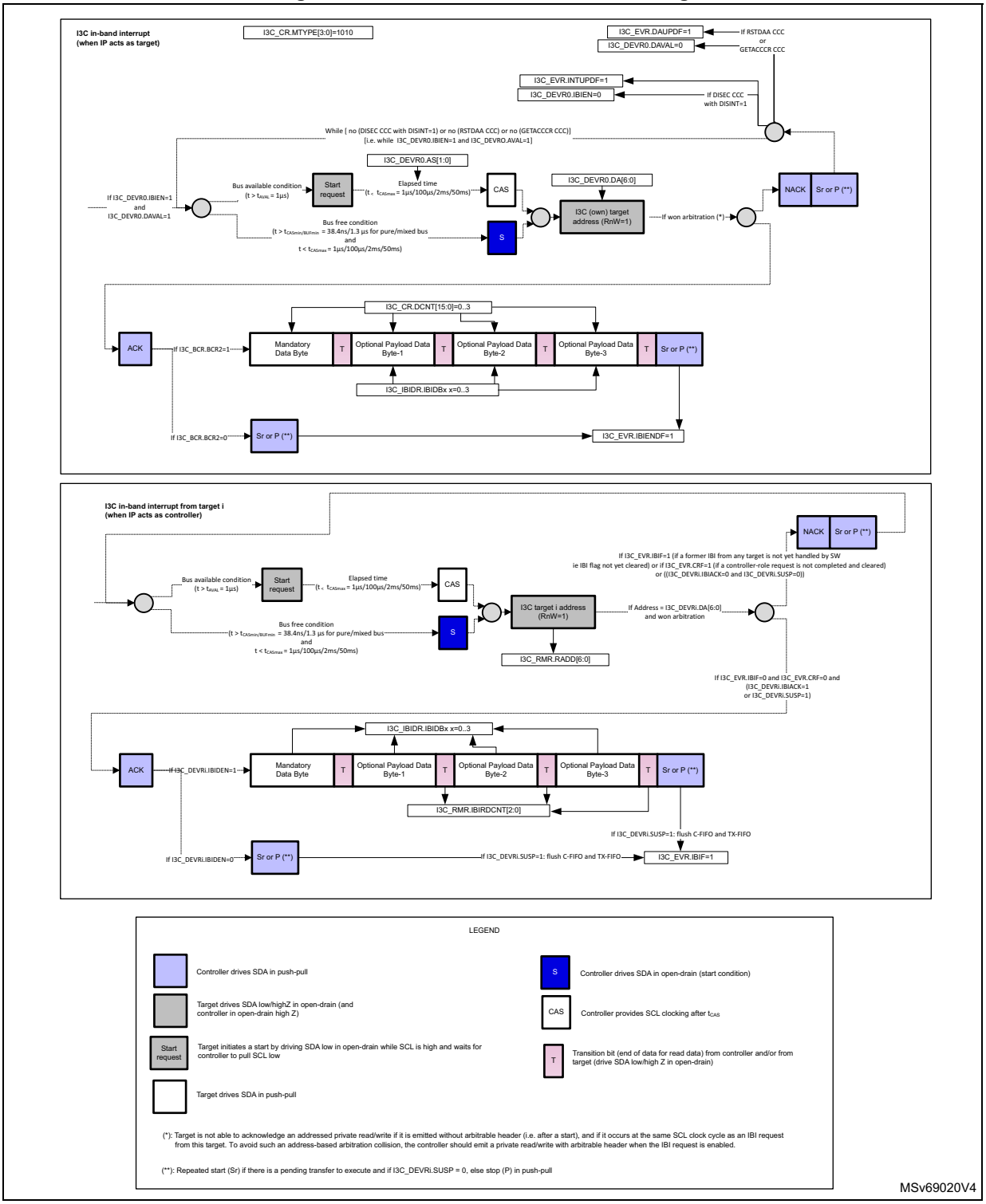
Figure 780. Legacy I²C read/write messages, as controller



66.9.13 I3C IBI transfer, as controller/target

Figure 781 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 781. IBI transfer, as controller/target



When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

Timing control and sensor acquisition timestamping when I3C controller

When the I3C peripheral acts a controller, it can use timing control and asynchronous mode 0 to timestamp the moment at which the target(s) acquired sampled data, and to time-correlate them.

To do this, the software must go through the following sequence:

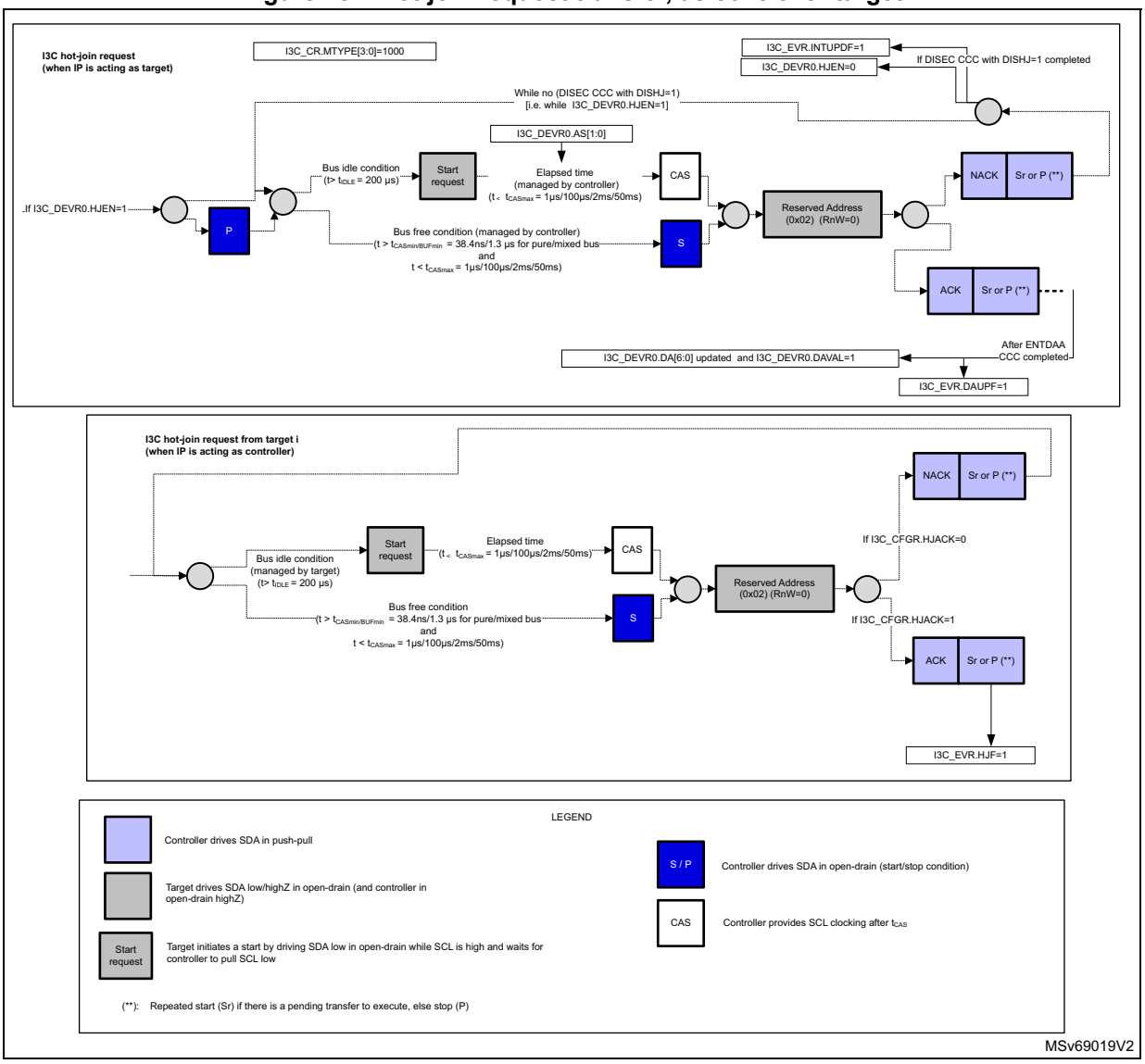
1. Allocate and configure a timer in capture mode, to be connected to the I3C generated event i3c_ibi_ack so that for each IBI acknowledge the counter of the timer can be captured.
2. Program a broadcast/direct SETXTIME CCC with sub-command byte 0xDT to enter asynchronous mode 0 on the target(s) side.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR register), reset and enable the timer to start the counter on the controller side.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF = 1 in the I3C_EVR register), and if the IBI is with timing info (if IBIDB0[7:5] = 100 in the I3C_IBIDR register), read the captured value of the timer, as C_TIM. Given that:
   a) C_C2 = 9 × (SCLL_PP[7:0] + 1 + SCLH_I3C[7:0] + 1) × T_{I3CCLK}
   b) T_C1 (16-bit value) = {IBIDB2, IBIDB1} in the I3C_IBIDR register
   c) T_C2 (8-bit value) = IBID3 in the I3C_IBIDR register

   Compute the timestamp of the sampled data received from the target, as follows (see MIPI spec formula for more details): C_TS = C_TIM - C_C2 x T_C1 / T_C2.
5. Repeat step *4*, for each received IBI.
6. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit asynchronous mode 0 and disable/deallocate the timer resource.

66.9.14 I3C hot-join request transfer, as controller/target

Figure 782 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

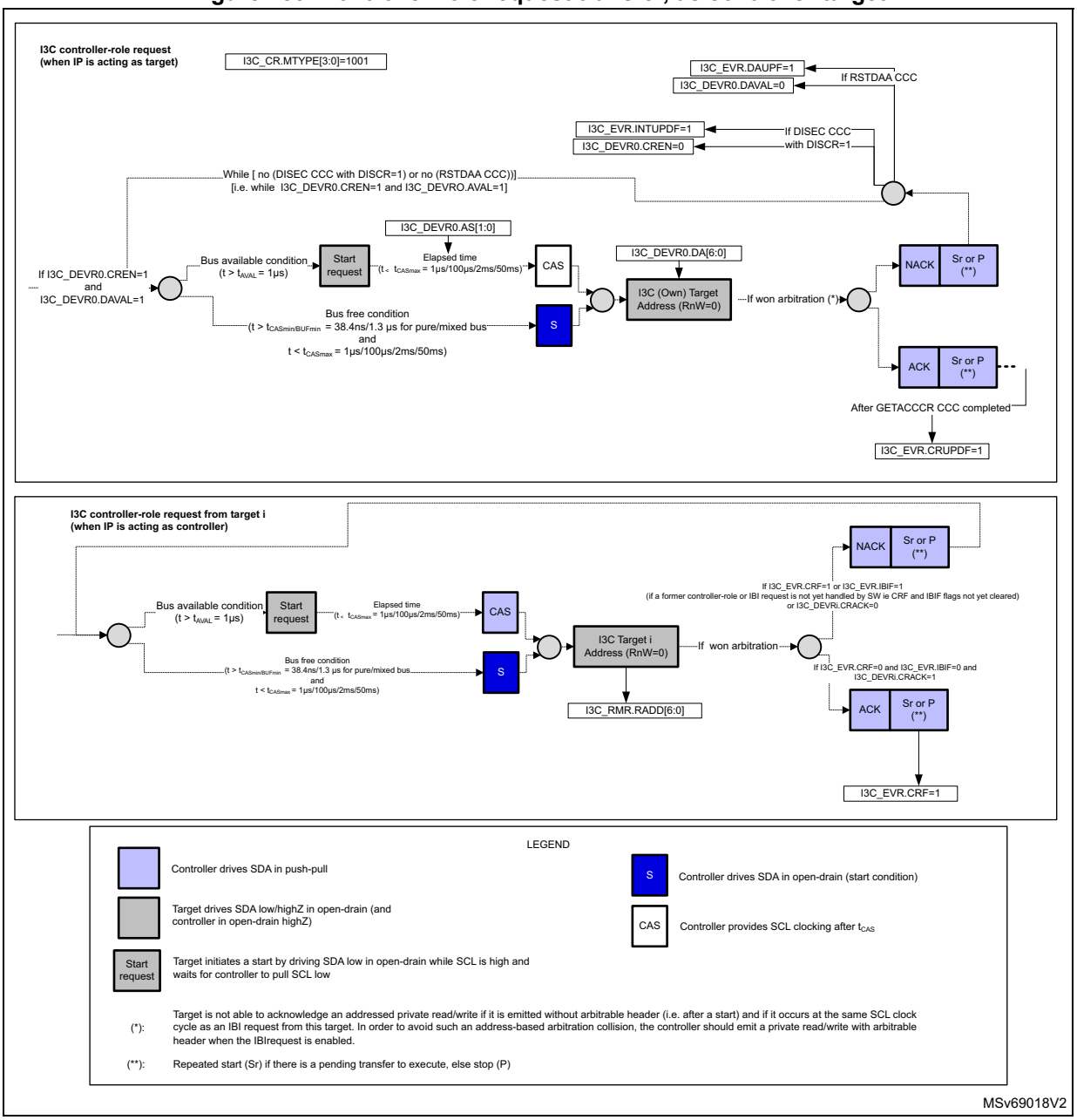
Figure 782. Hot-join request transfer, as controller/target



66.9.15 I3C controller-role request transfer, as controller/target

Figure 783 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 783. Controller-role request transfer, as controller/target



66.10 I3C FIFOs management, as controller

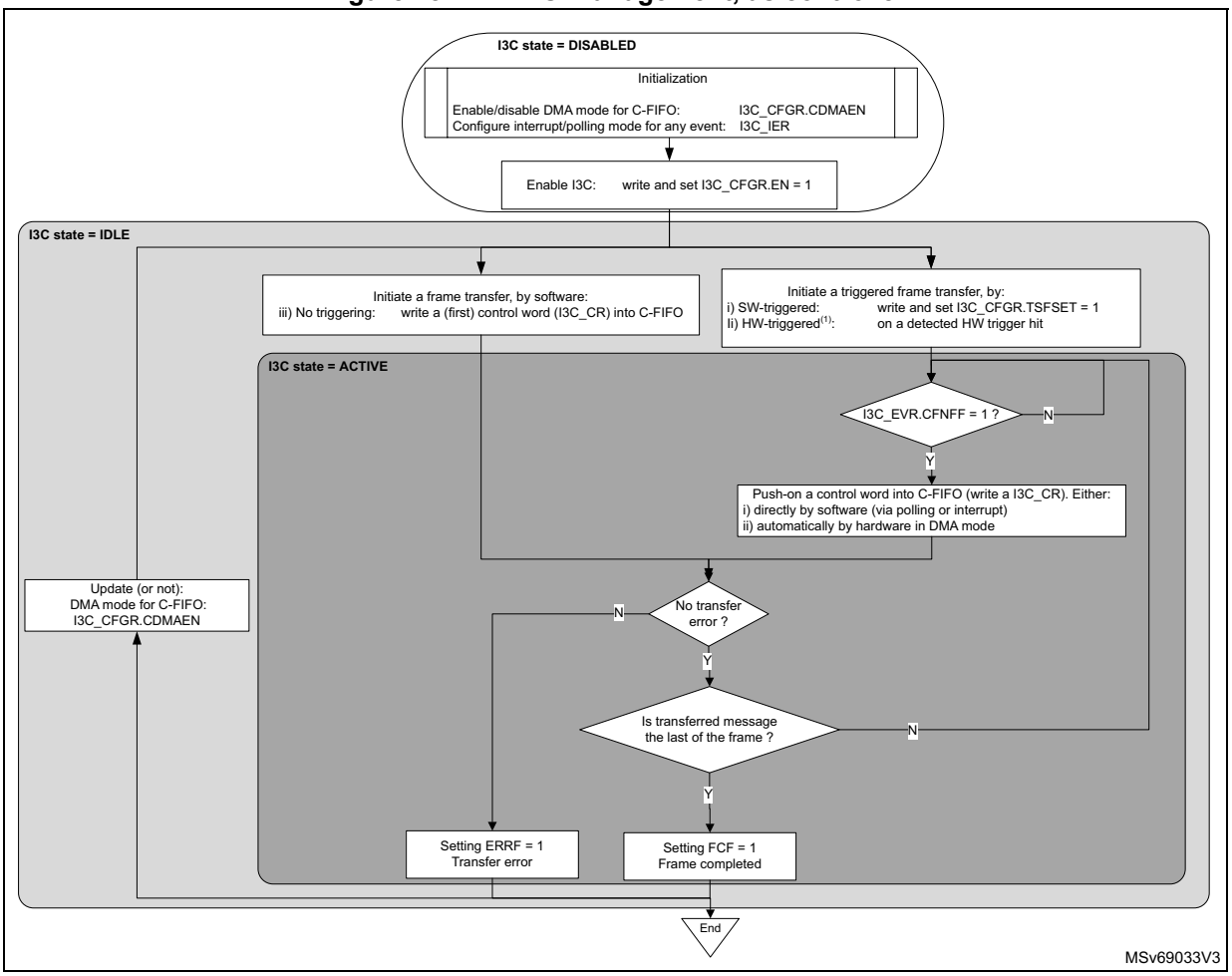
66.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 66.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 771*, *Figure 772*, *Figure 773*)
- direct read/write CCC (*Figure 773*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 778*)
- legacy I2C read/write (*Figure 780*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 784 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 784. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 66.3.5*.

Note: The software can disable the generation of FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 66.3) to stream a series of frames.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

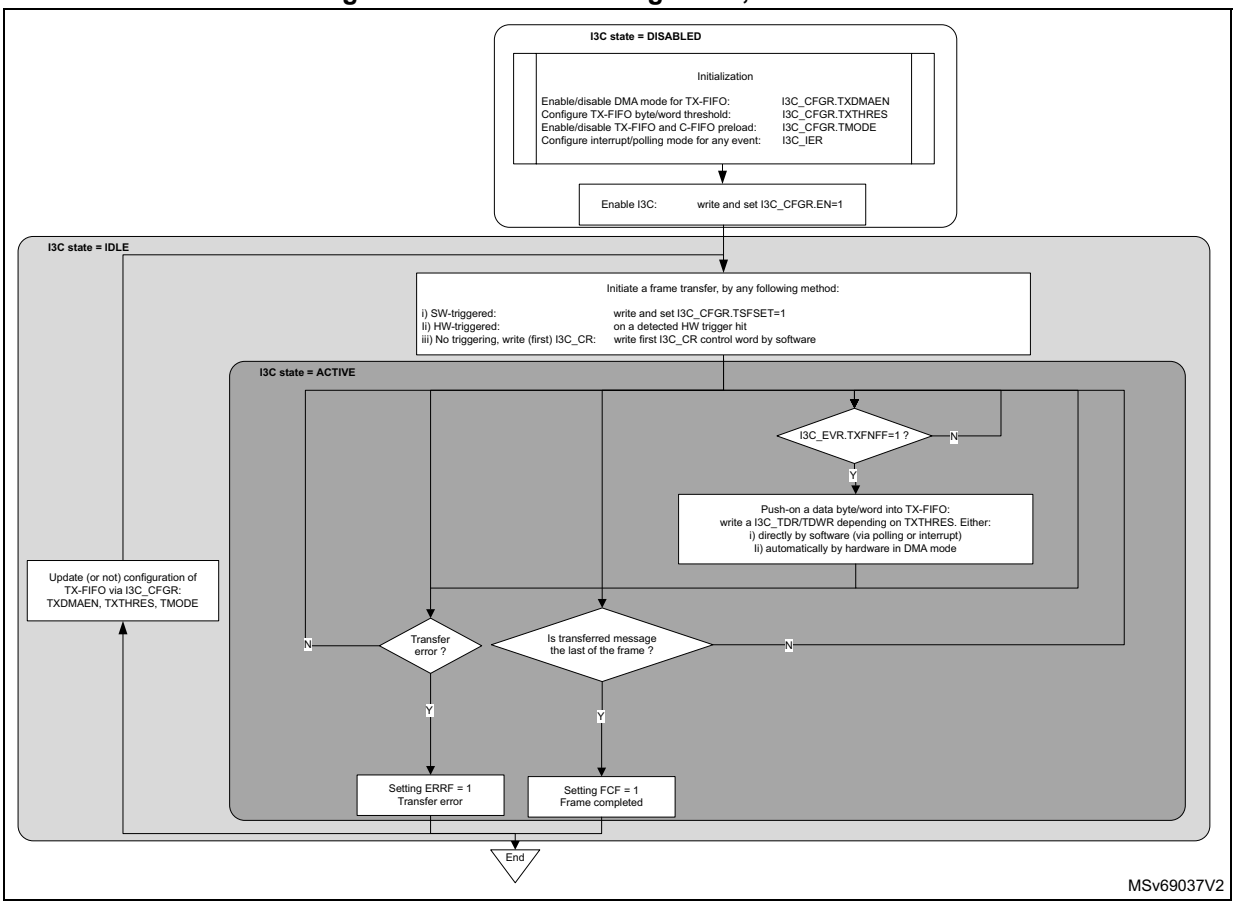
66.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 66.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 771*, *Figure 772*, *Figure 773*)
- private write (*Figure 778*)
- legacy I2C write (*Figure 780*)

Figure 785 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 785. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 667*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

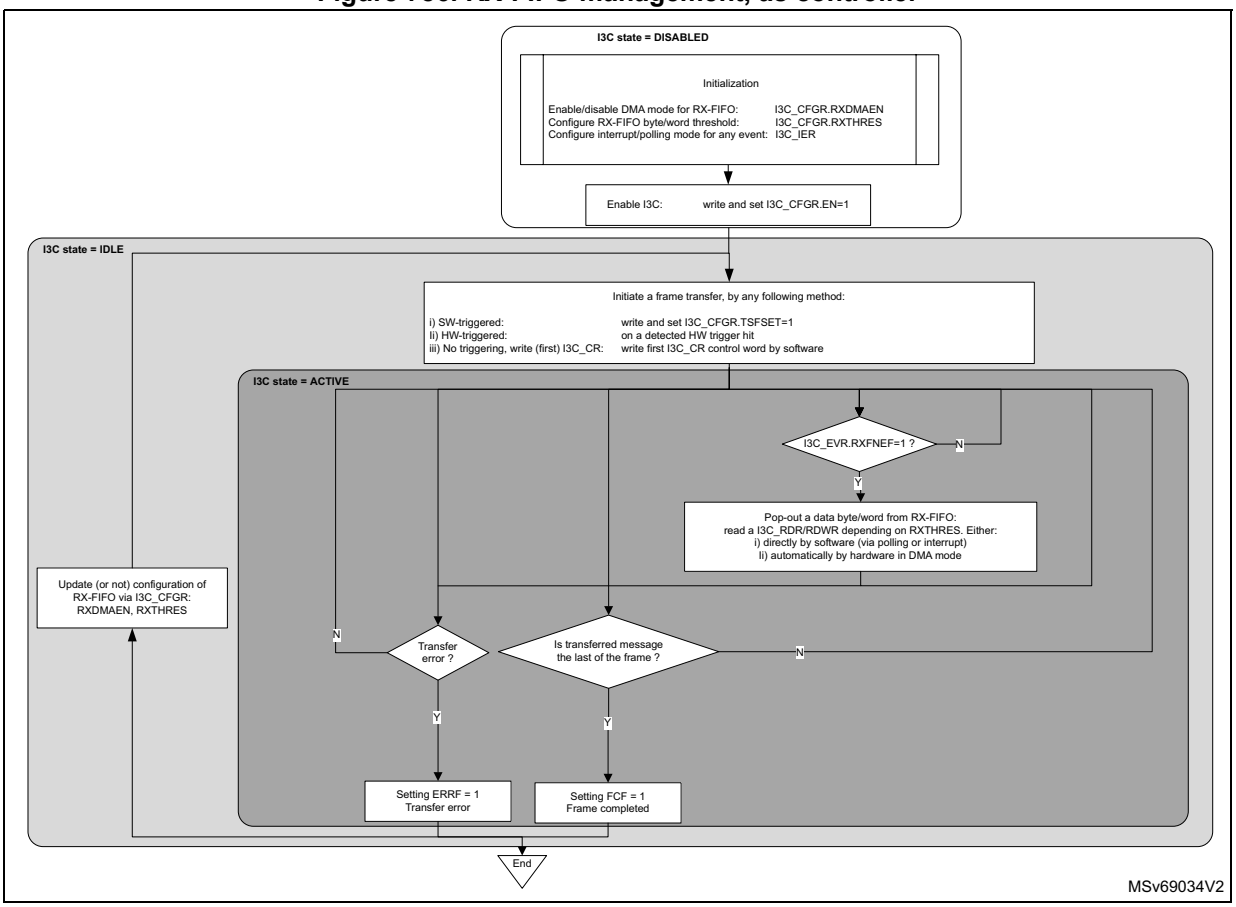
66.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 66.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 772*)
- direct CCC read (*Figure 771*), including direct RSTACT CCC read (*Figure 773*)
- private read (*Figure 778*)
- legacy I2C read (*Figure 780*)

Figure 786 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 786. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 66.10.4*.

66.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

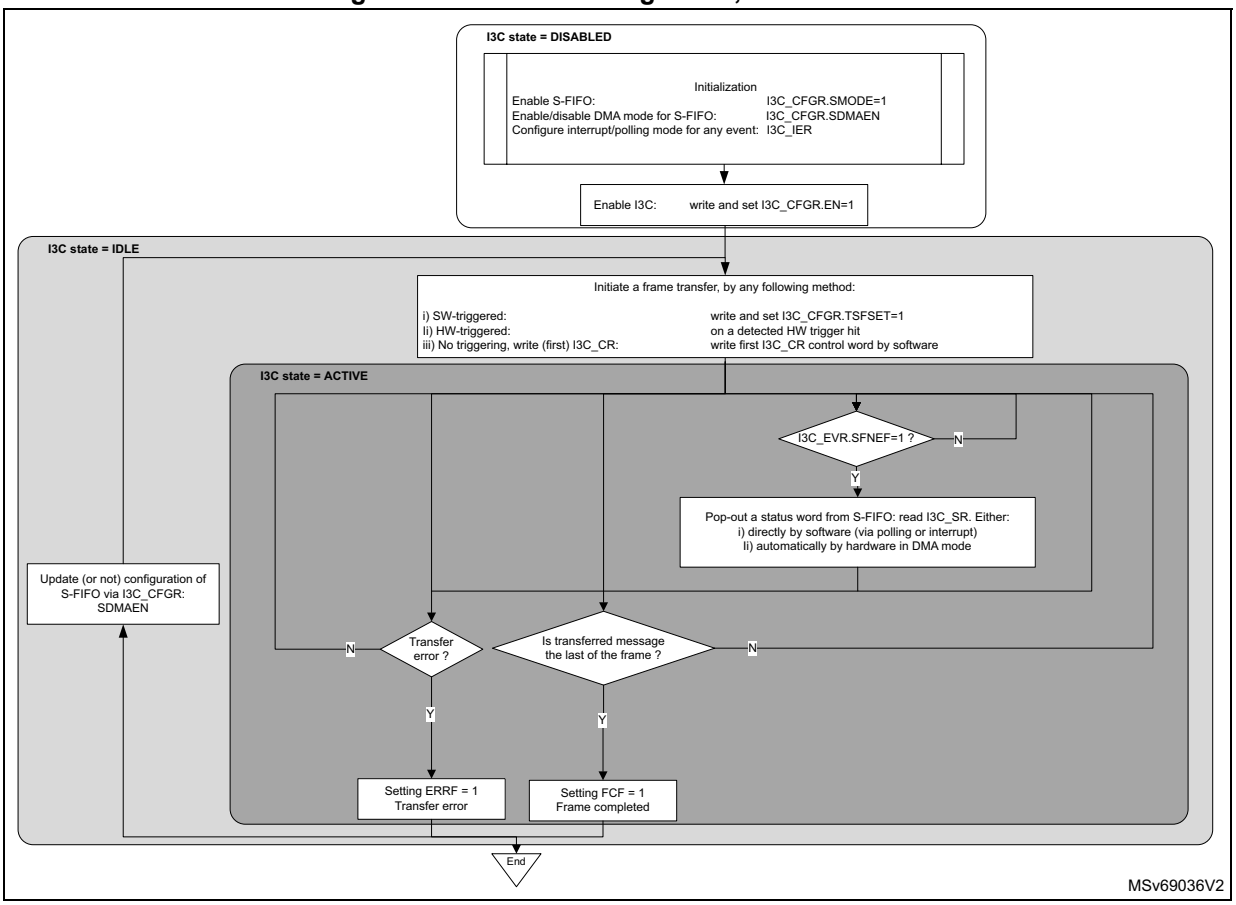
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 66.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 787* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 787. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

66.11 I3C FIFOs management, as target

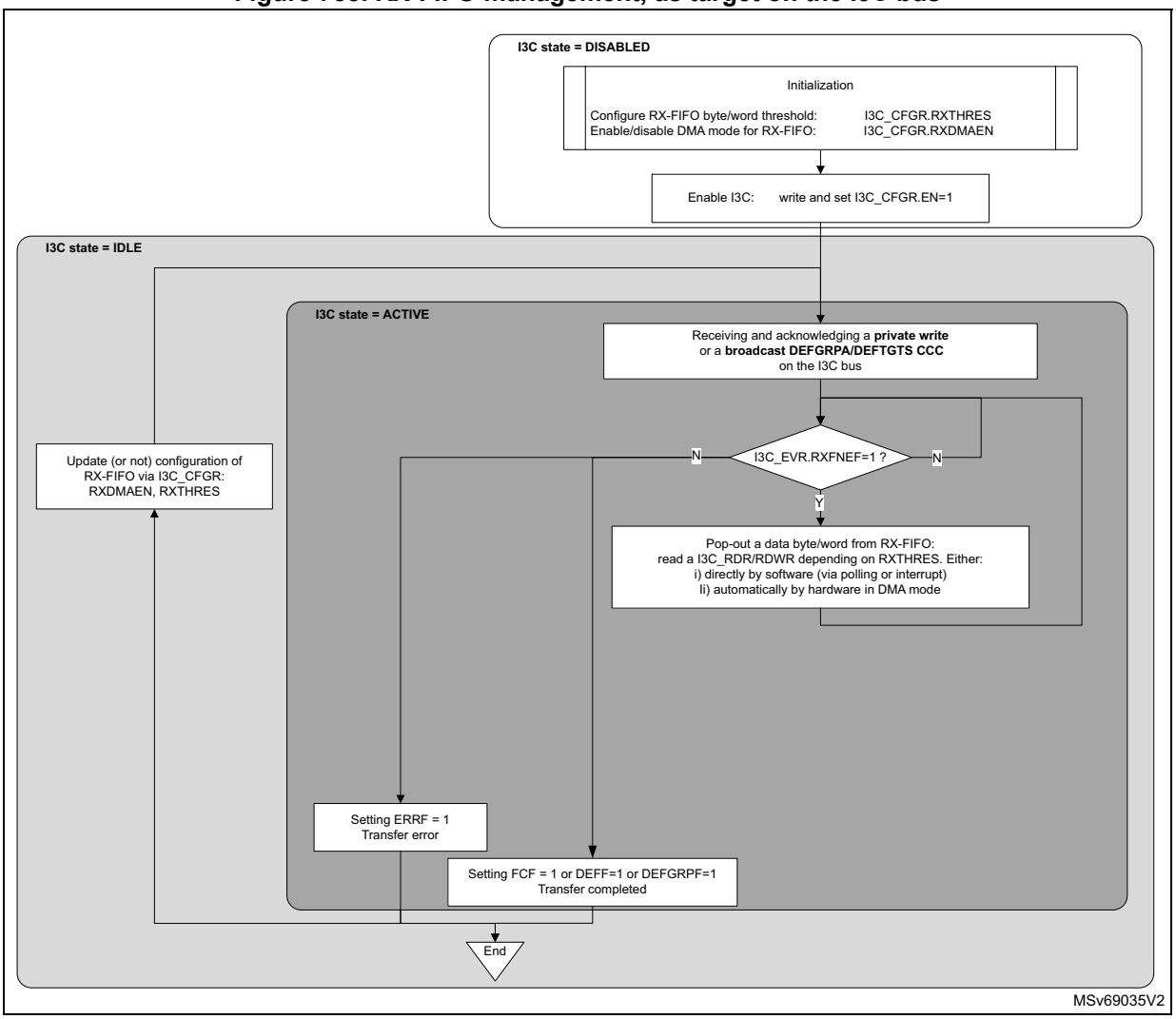
66.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 66.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 776*)
- broadcast DEFGRPA CCC (*Figure 777*)
- private write (*Figure 779*)

Figure 788 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received on the I3C bus, when the I3C peripheral acts as target.

Figure 788. RX-FIFO management, as target on the I3C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

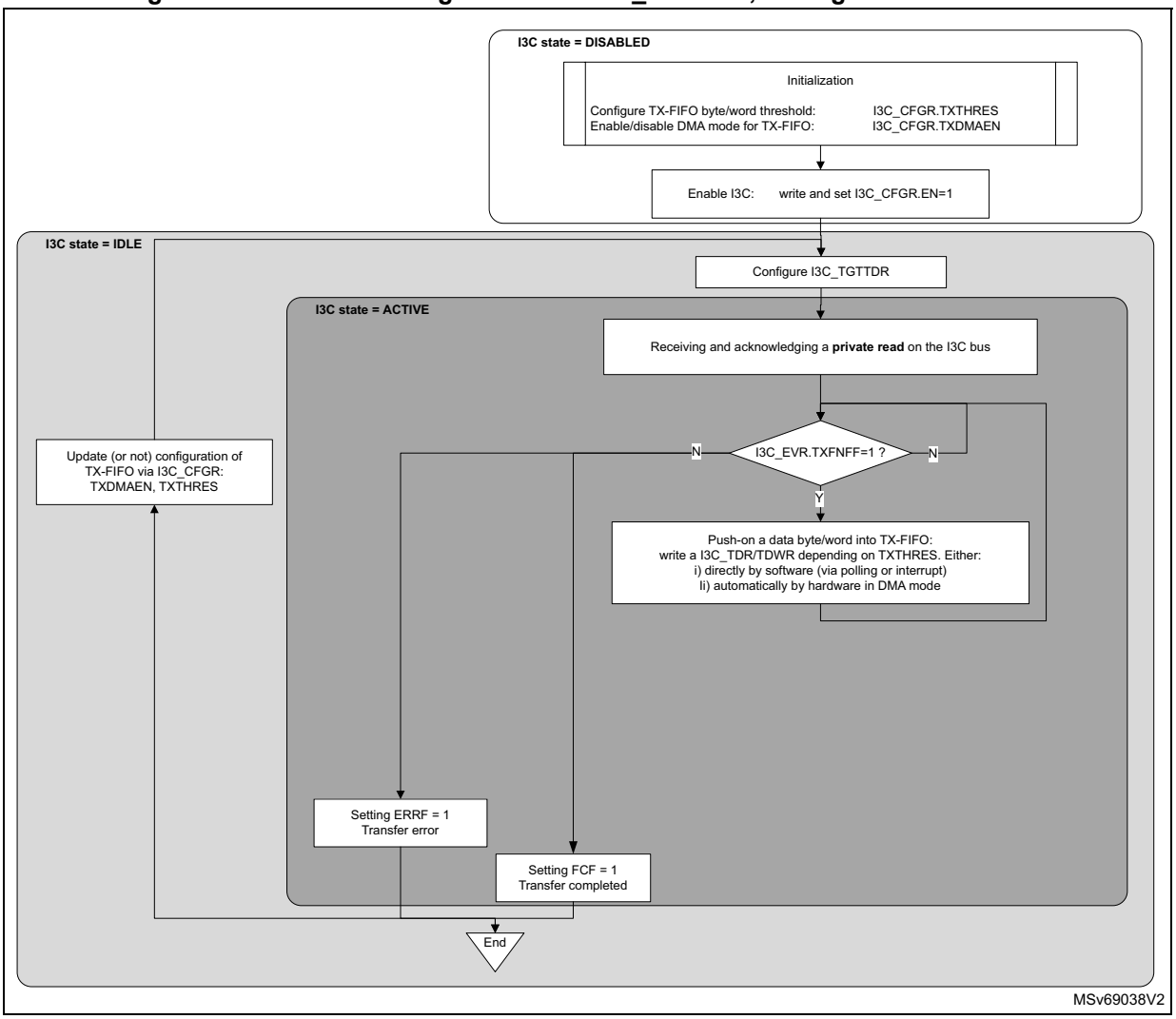
66.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 66.9*, the TX-FIFO is used only during a private read (*Figure 779*).

Figure 789 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 789. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus , the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 667).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

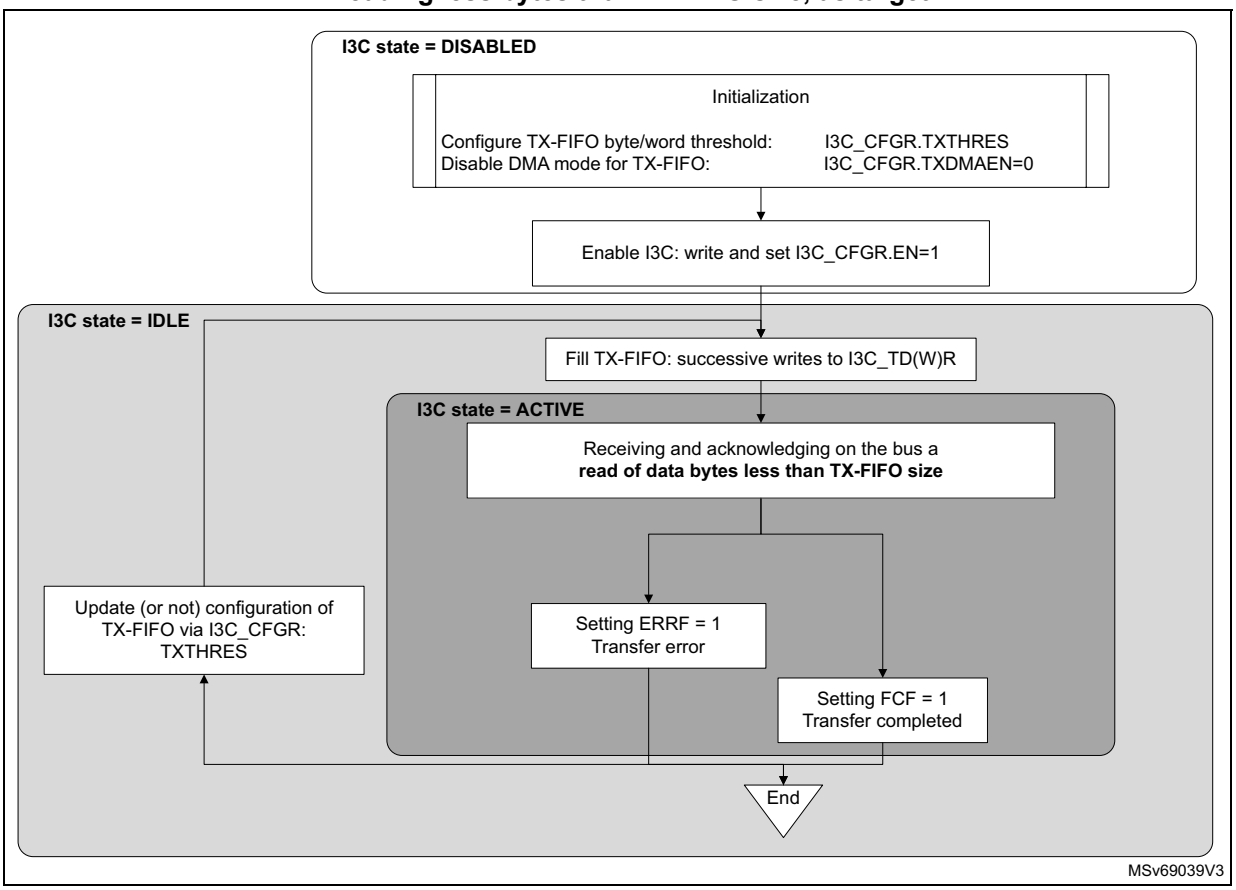
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 790*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 790. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



66.12 I3C error management

66.12.1 Controller error management

Table 676 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 676. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 676. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 676. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

66.12.2 Target error management

Table 677 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 677. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 677. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

66.13 I3C wake-up from low-power mode(s)

66.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 781*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 782*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 783*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.
7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the I3C bus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

66.14 I3C in low-power modes

Table 678. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop[1]	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode. Serial bus transfers can occur and can be assisted with an autonomous DMA for data transfers to/from memory, provided that the autonomous DMA can operate in Stop mode. Refer to *Section 66.13* for more details.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Table 665* and *Table 666* to know if Stop mode is supported, and which one.

66.15 I3C interrupts

66.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

66.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 66.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 676*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 676* and MIPI specification about escalation handling.

If the fields of the I3C_CFGR register are programmed with EXITPTRN= 0 and RSTPTRN=1, a target reset pattern is sent on the I3C bus after the header message, whatever the header is ACK-ed or NACK-ed. This can be used to generate and broadcast a target reset pattern, especially in the case of an I3C bus topology with a single I3C target in deep sleep. Alternatively, and more generally, a target reset pattern can be encapsulated within a RSTACT CCC command (refer to *Figure 773*).

0010: **private message** (refer to *Figure 778*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 771*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 780*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 782*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 783*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 781*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.

0: write message

1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.

When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).

Linear encoding up to 64 Kbytes - 1

0x0000: no data to transfer

0x0001: 1 byte

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

66.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 675*, *Figure 771*, *Figure 772*, *Figure 773*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 772*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 772).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 773).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

66.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH, FCFDIS)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w							rw		w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **FCFDIS**: FCF generation disable (when I3C acts as controller)

When the IP acts as controller, the generation of the FCF flag in the register I3C_EVR can be disabled. The software can avoid clearing this flag between two generated I3C frames (by writing 1 into CFCF in the I3C_CEVR register), especially in hardware-triggered autonomous mode.

0: enable the generation of the flag FCF in the I3C_EVR register

1: disable the generation of the flag FCF in the I3C_EVR register

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 66.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Additionally, if RSTPTRN = 1 and EXITPTRN = 0, a reset pattern is sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register), whatever the message header is ACK-ed or NACK-ed.

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).
- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.
- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

66.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

66.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

66.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

66.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

66.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

66.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

66.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

66.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

66.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.
								r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

66.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

66.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

66.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

66.16.16 I3C masked interrupt status register (I3C_MISR)

Address offset: 0x05C

Reset value: 0x0000 0000

This is a read register, containing the masked interrupt status for each individual event flag bit. Each bit is the logical AND of the corresponding interrupt enable bit of the I3C_IER register and the event flag bit of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPMIS**: DEFGRPA CCC masked interrupt status (when the I3C acts as target)

Bit 30 **DEFMIS**: DEFTGTS CCC masked interrupt status (when the I3C acts as target)

Bit 29 **INTUPDMIS**: ENEC/DISEC CCC masked interrupt status (when the I3C acts as target)

Bit 28 **ASUPDMIS**: ENTASx CCC masked interrupt status (when the I3C acts as target)

Bit 27 **RSTMIS**: reset pattern masked interrupt status (when the I3C acts as target)

Bit 26 **MRLUPDMIS**: SETMRL CCC masked interrupt status (when the I3C acts as target)

Bit 25 **MWLUPDMIS**: SETMWL CCC masked interrupt status (when the I3C acts as target)

Bit 24 **DAUPDMIS**: ENTDAA/RSTDAA/SETNEWDA CCC masked interrupt status (when the I3C acts as target)

Bit 23 **STAMIS**: format 1 GETSTATUS CCC masked interrupt status (when the I3C acts as target)

Bit 22 **GETMIS**: GETxxx CCC masked interrupt status, except GETSTATUS of format 1 (when the I3C acts as target)

Bit 21 **WKPMIS**: Wake-up/missed start masked interrupt status (when the I3C acts as target)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJMIS**: Hot-join masked interrupt status (when the I3C acts as controller)

Bit 18 **CRUPDMIS**: Controller-role update masked interrupt status (when the I3C acts as target)

Bit 17 **CRMIS**: Controller-role request masked interrupt status (when the I3C acts as controller)

Bit 16 **IBIENDMIS**: IBI end masked interrupt status (when the I3C acts as target)

Bit 15 **IBIMIS**: IBI request masked interrupt status (when the I3C acts as controller)

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRMIS**: error masked interrupt status (whatever the I3C acts as controller/target)

Bit 10 **RXTGTENDMIS**: target-initiated read end masked interrupt status (when the I3C acts as controller)

Bit 9 **FCMIS**: frame complete masked interrupt status (whatever the I3C acts as controller/target)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEMIS**: RX-FIFO not empty masked interrupt status (whatever the I3C acts as controller/target)

Bit 4 **TXFNFMIS**: TX-FIFO not full masked interrupt status (whatever the I3C acts as controller/target)

Bit 3 **SFNEMIS**: S-FIFO not empty masked interrupt status (when the I3C acts as controller)

Bit 2 **CFNFMIS**: C-FIFO not full masked interrupt status (when the I3C acts as controller)

Bits 1:0 Reserved, must be kept at reset value.

66.16.17 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.

This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:

– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.

– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

– the software writes and sets SYSRST = 1 in the RCC_GRSTCSETR register, when the device is controlled by a Cortex®-A.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

66.16.18 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

66.16.19 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

66.16.20 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MWL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

66.16.21 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

66.16.22 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SDA_HD[1:0]		Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
		rw	rw						rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SDA_HD[1:0]**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD[1:0] + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

66.16.23 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bit 7 Reserved, must be kept at reset value.

Bit 6 **STALLL:** Controller clock stall enable in the address ACK/NACK phase (before the ninth bit) of a legacy I2C read/write message. This allows the I2C target to prepare the next data byte to send/receive.

0: no stall

1: stall enabled

Bit 5 **STALLS:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C write message. This allows the I2C target to prepare the next data byte to receive.

0: no stall

1: stall enabled

Bit 4 **STALLR:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C read message. This allows the I2C target to prepare the next data byte to send.

0: no stall

1: stall enabled

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

66.16.24 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable

0: I3C target (no controller capable)

1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

66.16.25 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device
Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

66.16.26 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

66.16.27 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

66.16.28 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

66.16.29 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).
This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector
This field is set as 0 (vendor fixed value).
This field represents bit[32] of the 48-bit provisioned ID.
Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.
This field represents bits[15:12] of the 48-bit provisioned ID.
Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

66.16.30 I3C hardware configuration register (I3C_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0004 2222

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBT[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSIZE[3:0]				TSIZE[3:0]				SSIZE[3:0]				CSIZE[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NBT[3:0]**: number of targets whose the IBI/controller-role request can be managed in parallel by this I3C IP, when acting as controller

This value drives the maximum value x of the I3C_DEVRx.

Bits 15:12 **RSIZE[3:0]**: RX-FIFO size, in multiple of 4 bytes.
Note: A RX-FIFO element is a (received) byte.

Bits 11:8 **TSIZE[3:0]**: TX-FIFO size, in multiple of 4 bytes.
Note: A TX-FIFO element is a (transmitted) byte.

Bits 7:4 **SSIZE[3:0]**: S-FIFO size, in multiple of (32-bit) words.
Note: A S-FIFO element is a (status) word.

Bits 3:0 **CSIZE[3:0]**: C-FIFO size, in multiple of (32-bit) words.
Note: A C-FIFO element is a (control) word.

66.16.31 I3C version register (I3C_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** I3C major revision

Bits 3:0 **MINREV[3:0]:** I3C minor revision

66.16.32 I3C identification register (I3C_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0081

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: I3C identification

66.16.33 I3C size identification register (I3C_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** I3C address space size identification (1 Kbyte)

66.16.34 I3C register map

Table 679. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW	DCNT[15:0]																
										CCC[7:0]																								
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0								0	0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 679. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									

Table 679. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x05C	I3C_MISR	GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS	IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0				0	0	0	0		
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]							SCLL_OD[7:0]									SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	SDA_HD		Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value			0	0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR		Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			

Table 679. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												
0x3F0	I3C_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBS[3:0]				RSIZE[3:0]				TSIZE[3:0]				SSIZE[3:0]				CSIZE[3:0]			
	Reset value													0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0x3F4	I3C_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0x3F8	I3C_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0x3FC	I3C_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

67 Universal synchronous/asynchronous receiver transmitter (USART/UART)

67.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

67.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

67.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

67.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 680. Instance implementation on STM32MP21xx

Instance	STM32MP21xx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
UART4	Basic
UART5	Basic
UART7	Basic
LPUART1	Low-power

Table 681. USART/LPUART features

Modes/features(1)	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X(2)
Continuous communication using DMA	X	X	X

Table 681. USART/LPUART features (continued)

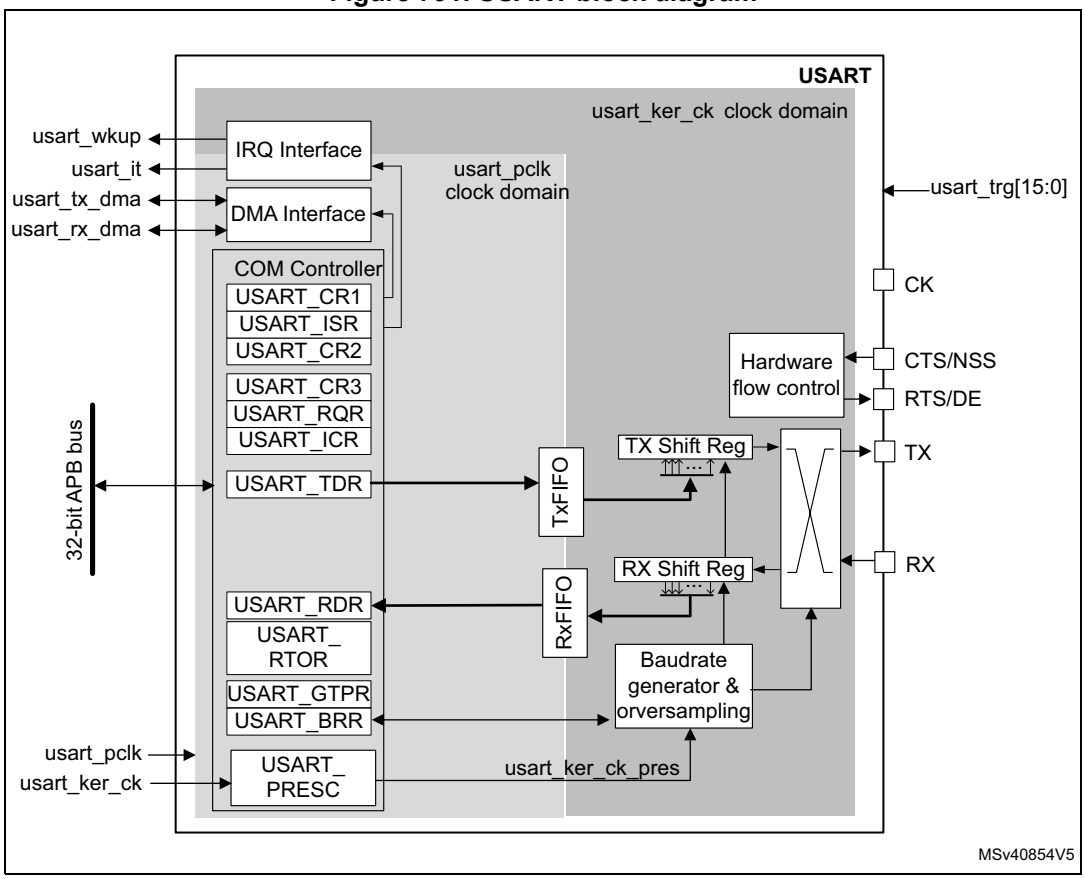
Modes/features[1]	Full feature	Basic feature	Low-power feature
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual-clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	-
USART data length	7, 8, and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	64		8
Wake-up from low-power mode	X[3]	X[3]	X[3]
Autonomous mode	X	X	X

1. “X” = supported, “-” = not supported.
2. Only the RS232 CTS flow control is supported.
3. Wake-up supported from Stop 1, Stop 2, Low-power stop 1, and Low-power stop 2 modes.

67.5 USART functional description

67.5.1 USART block diagram

Figure 791. USART block diagram



67.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 682* and *Table 683* for the list of USART input/output pins and internal signals.

Table 682. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 683. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request
usart_trg[15:0]	Input	USART triggers.

Description of USART interconnections

Table 684. USART interconnection (USART1/2/3/6 and UART4/5/7)

Signal name	Source
usart_trg0	-
usart_trg1	-
usart_trg2	-
usart_trg3	-
usart_trg4	exti4 (from EXIT2)
usart_trg5	exti5 (from EXTI2)
usart_trg6	-
usart_trg7	lptim3_ch1
usart_trg8	lptim4_ch1
usart_trg9	lptim5_out
usart_trg10	rtc_alra_trg
usart_trg11	rtc_wut_trg
usart_trg12	exti6 (from EXTI1)
usart_trg13	exti7 (from EXTI1)
usart_trg14	lptim1_ch1
usart_trg15	lptim2_ch1

67.5.3 USART clocks

The simplified block diagram given in *Section 67.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

67.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 792*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

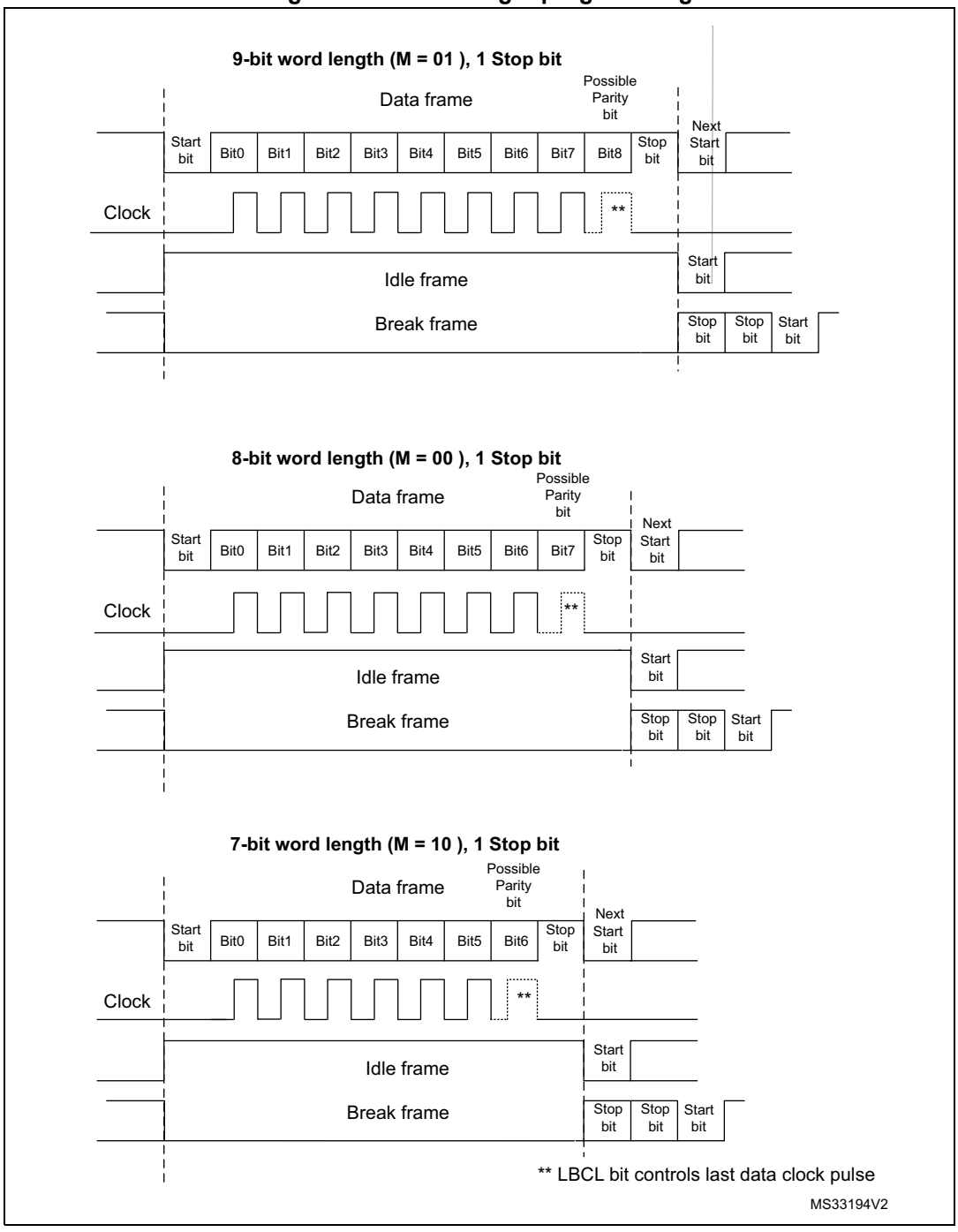
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 792. Word length programming



67.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

67.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

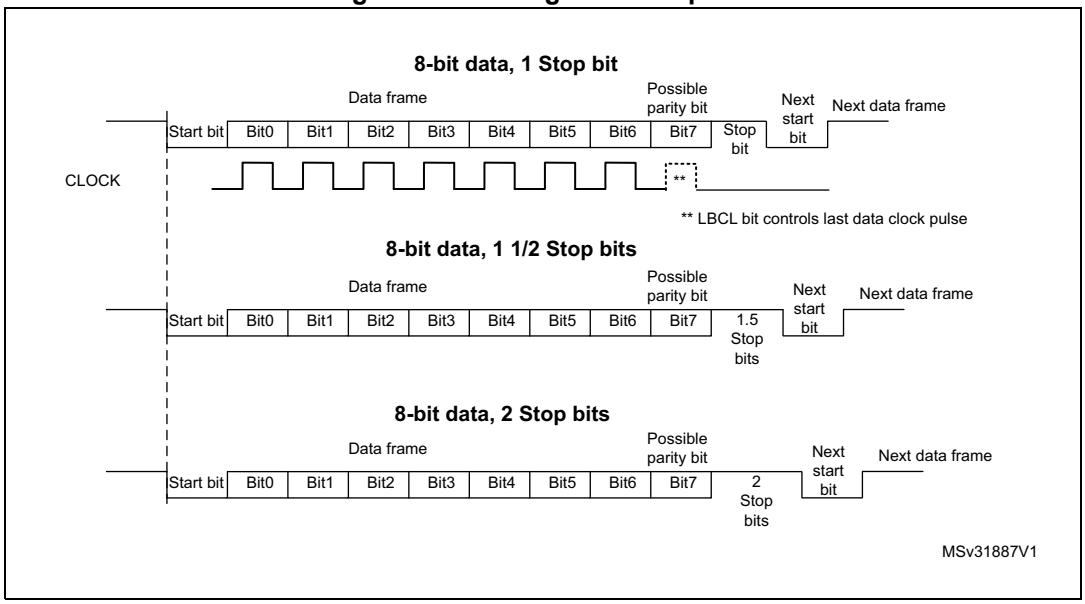
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 67.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 793. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 67.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

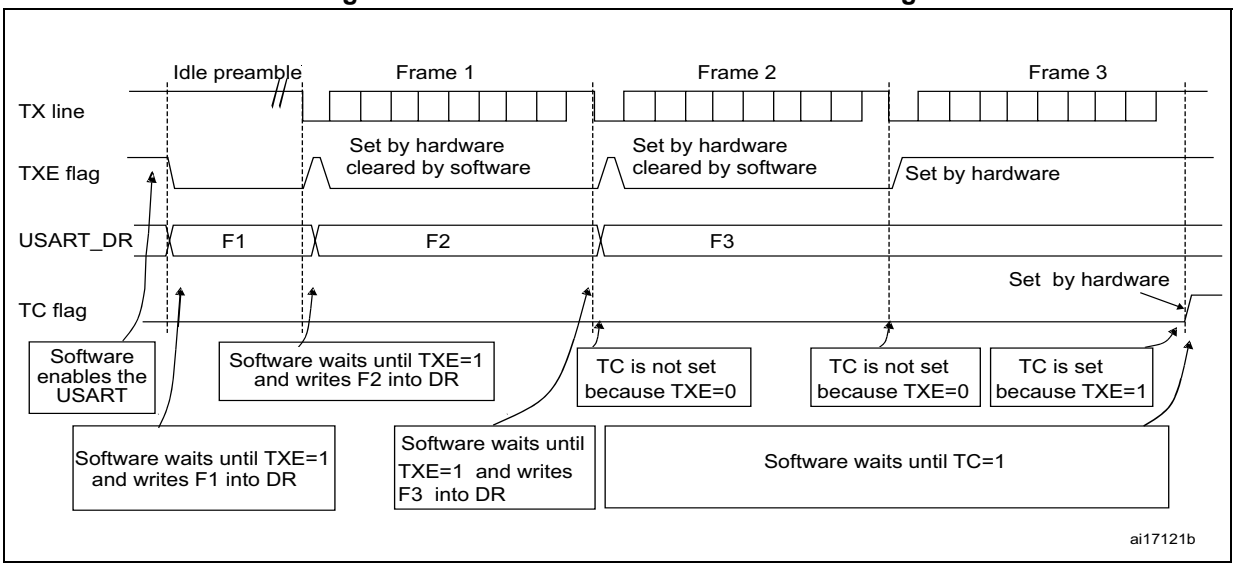
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 794*).

Figure 794. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 792*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

67.5.7 USART receiver

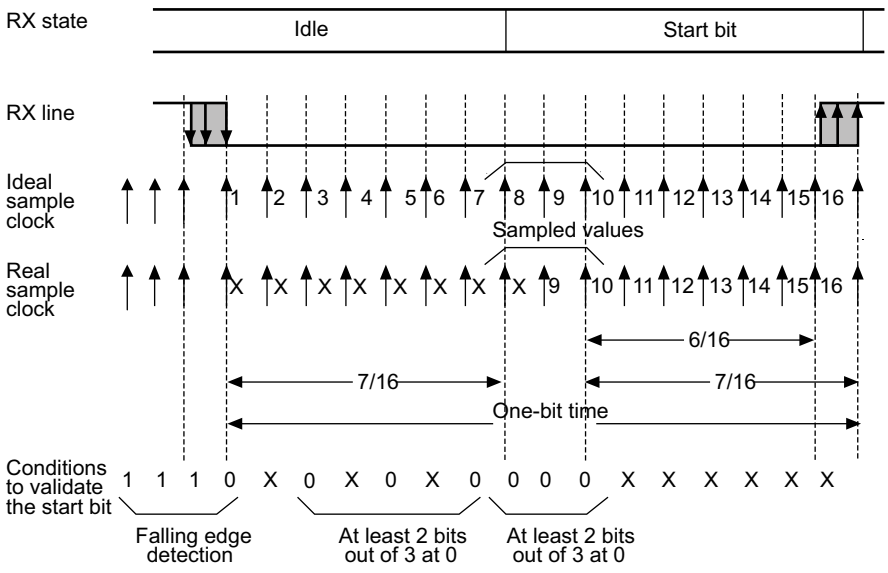
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 795. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 67.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

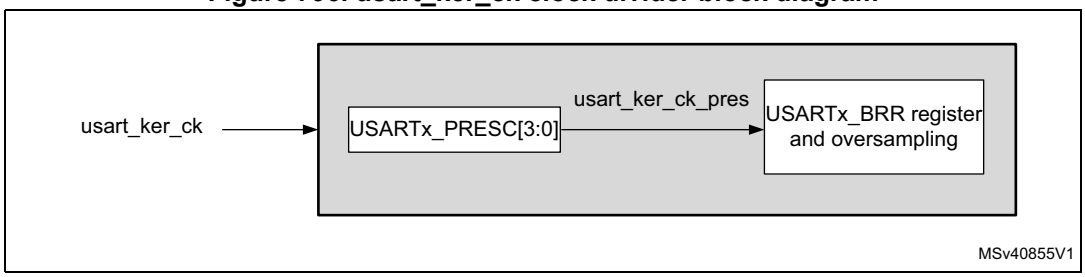
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 796. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 797* and *Figure 798*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 67.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 685*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 67.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 797. Data sampling when oversampling by 16

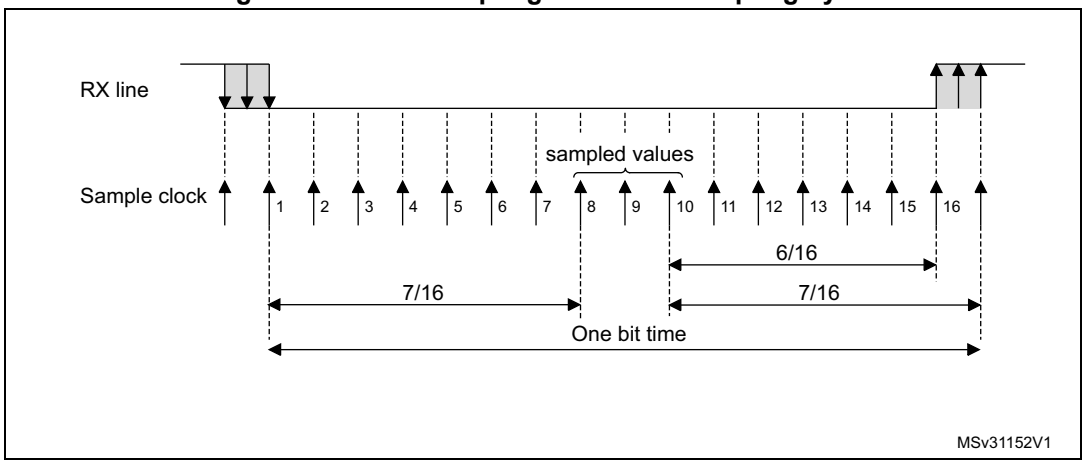


Figure 798. Data sampling when oversampling by 8

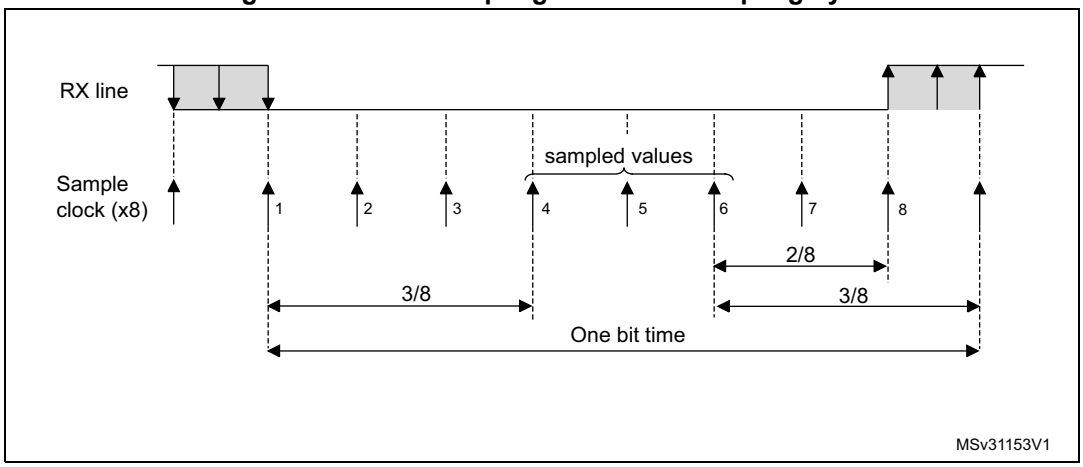


Table 685. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 67.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

67.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

67.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 686*, *Table 687*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 686. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 687. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 686 and Table 687 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

67.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

67.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

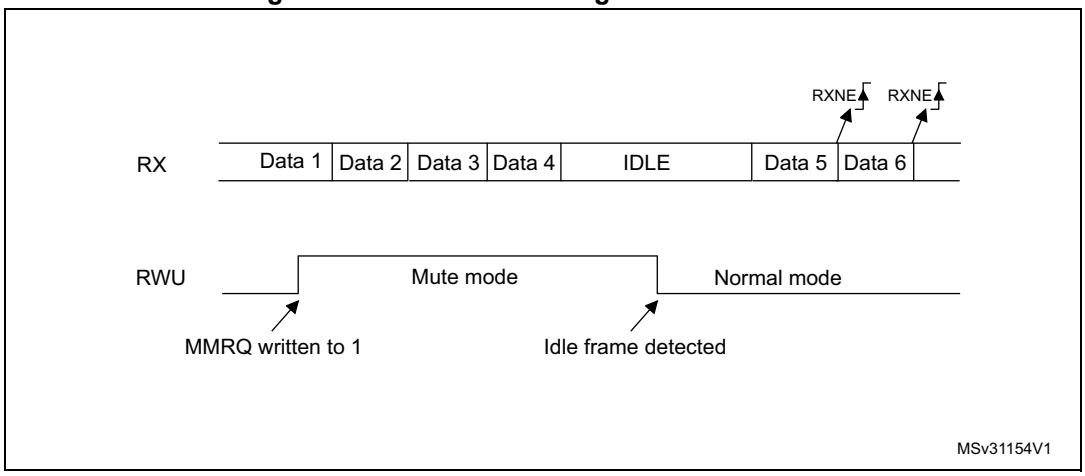
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 799*.

Figure 799. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

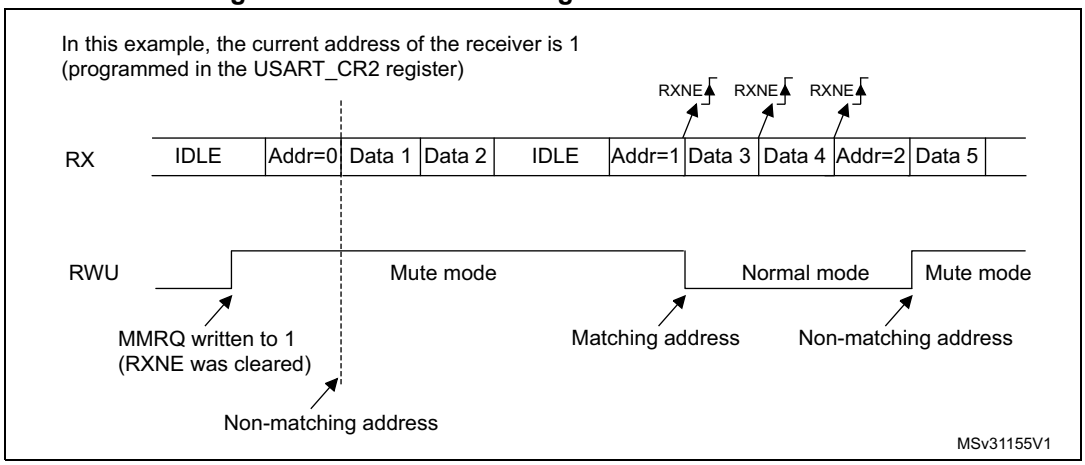
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 800*.

Figure 800. Mute mode using address mark detection



67.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

67.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 688*.

Table 688. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

67.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 67.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 67.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 801*.

Examples of break frames are given in *Figure 802*.

Figure 801. Break detection in LIN mode (11-bit break length - LBDL bit is set)

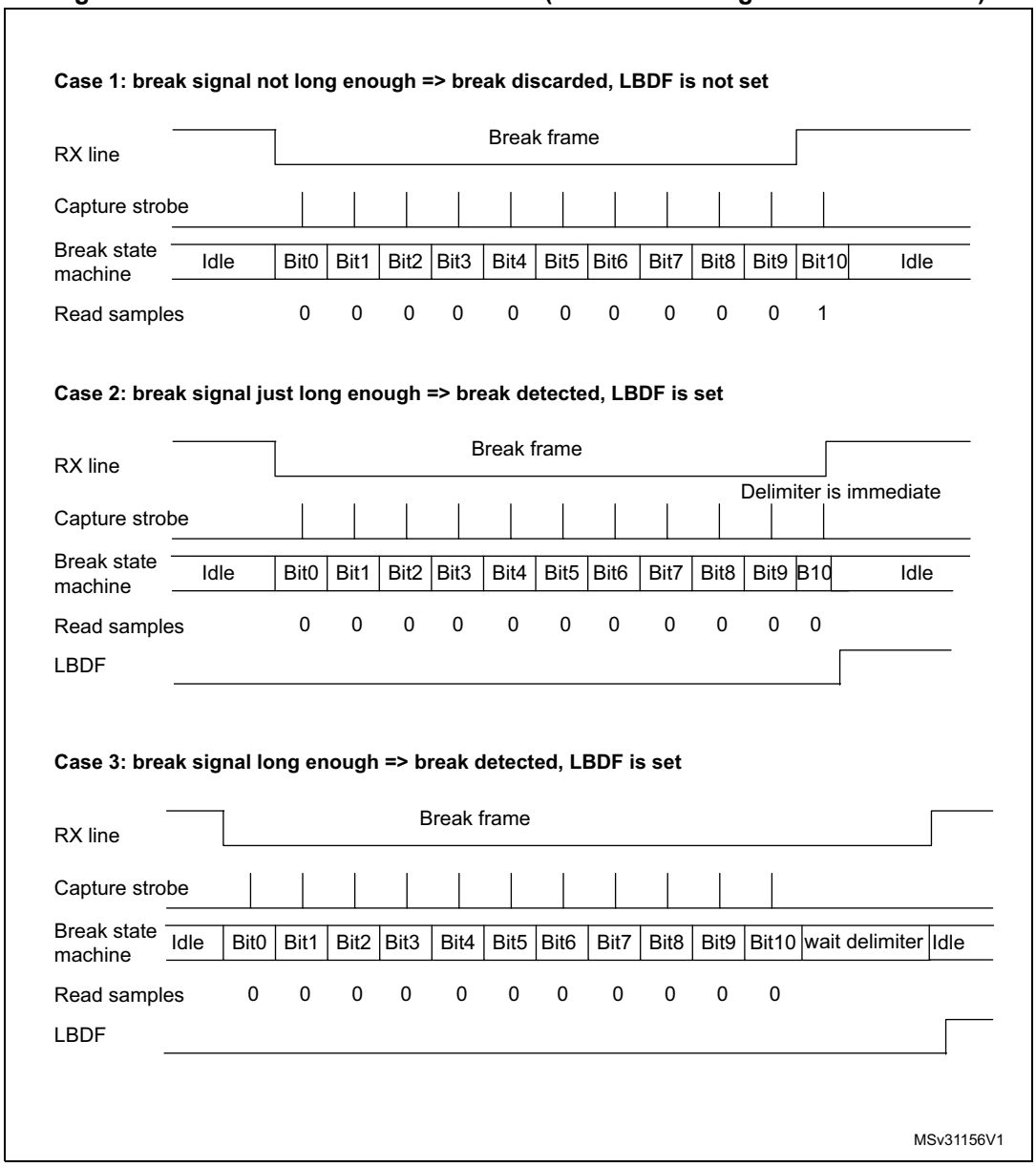
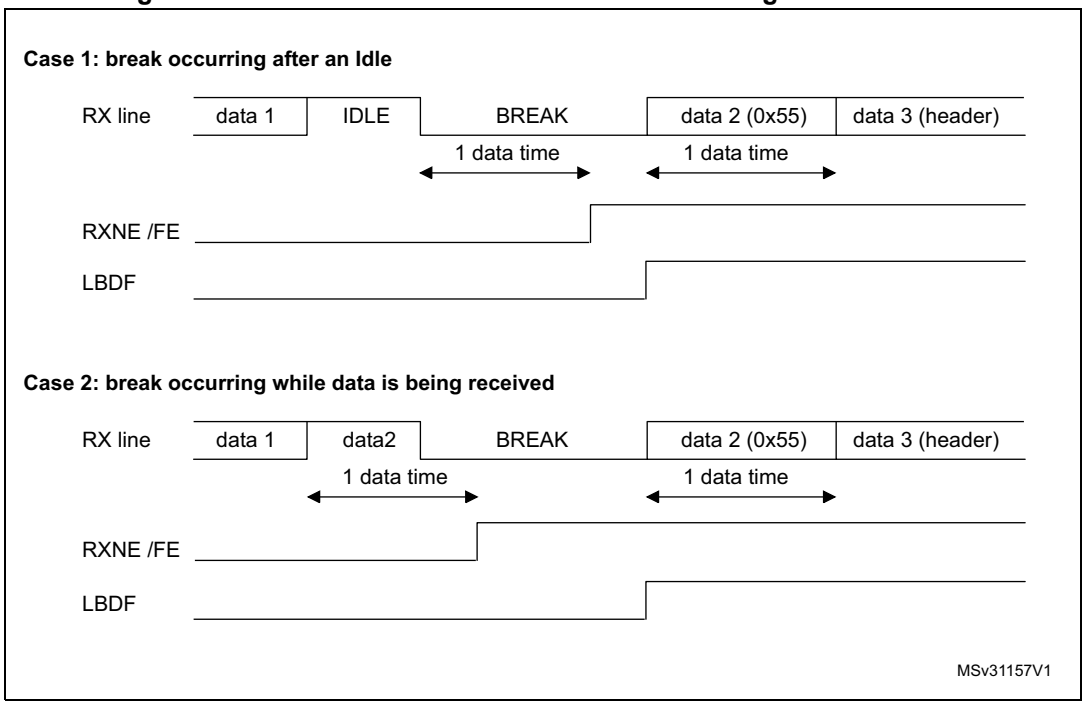


Figure 802. Break detection in LIN mode vs. Framing error detection



67.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 803*, *Figure 804* and *Figure 805*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 803. USART example of synchronous master transmission

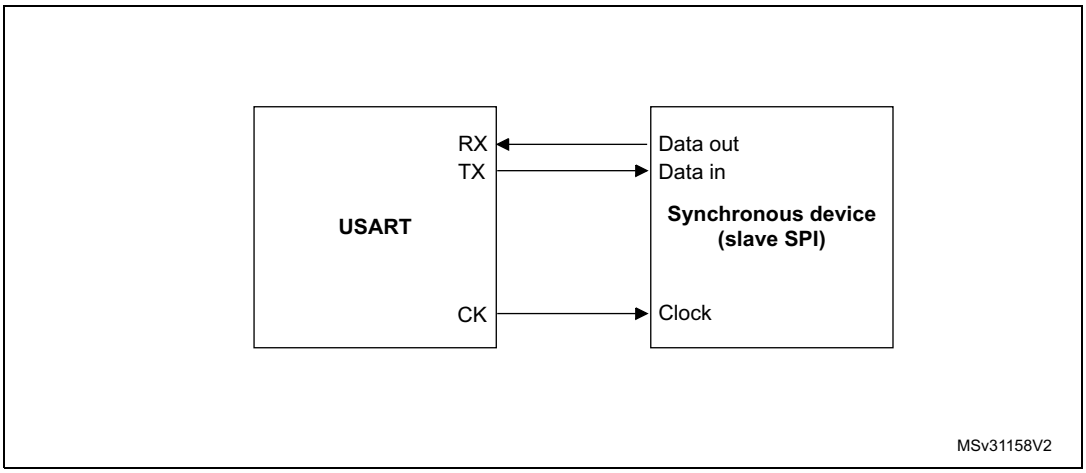


Figure 804. USART data clock timing diagram in synchronous master mode (M bits = 00)

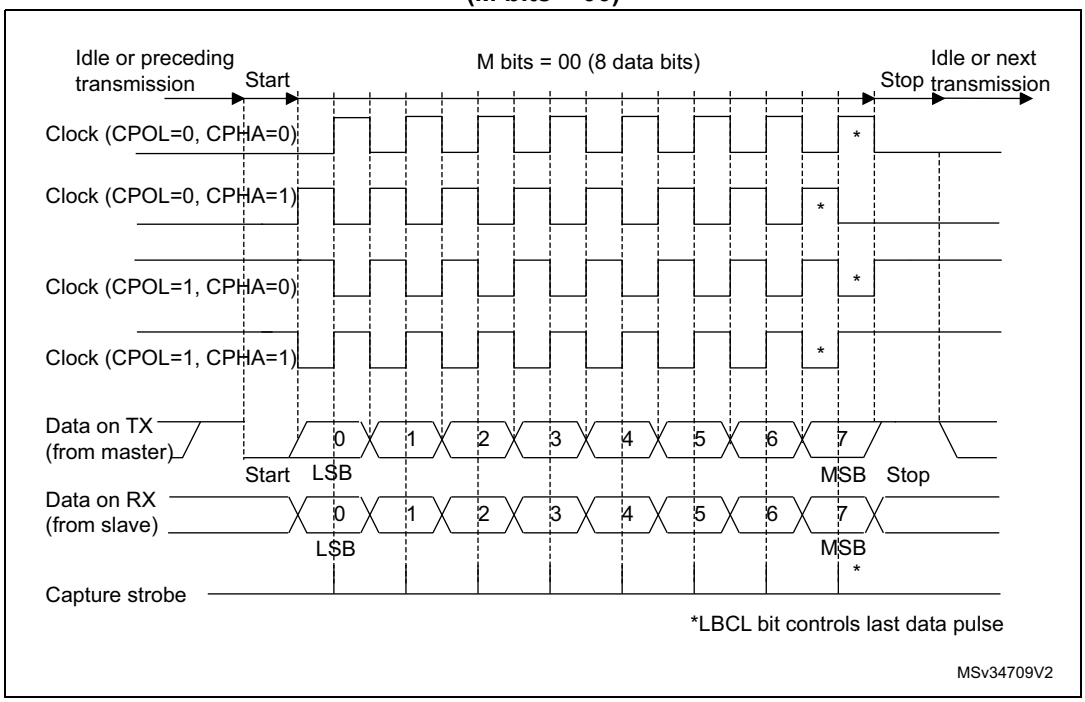
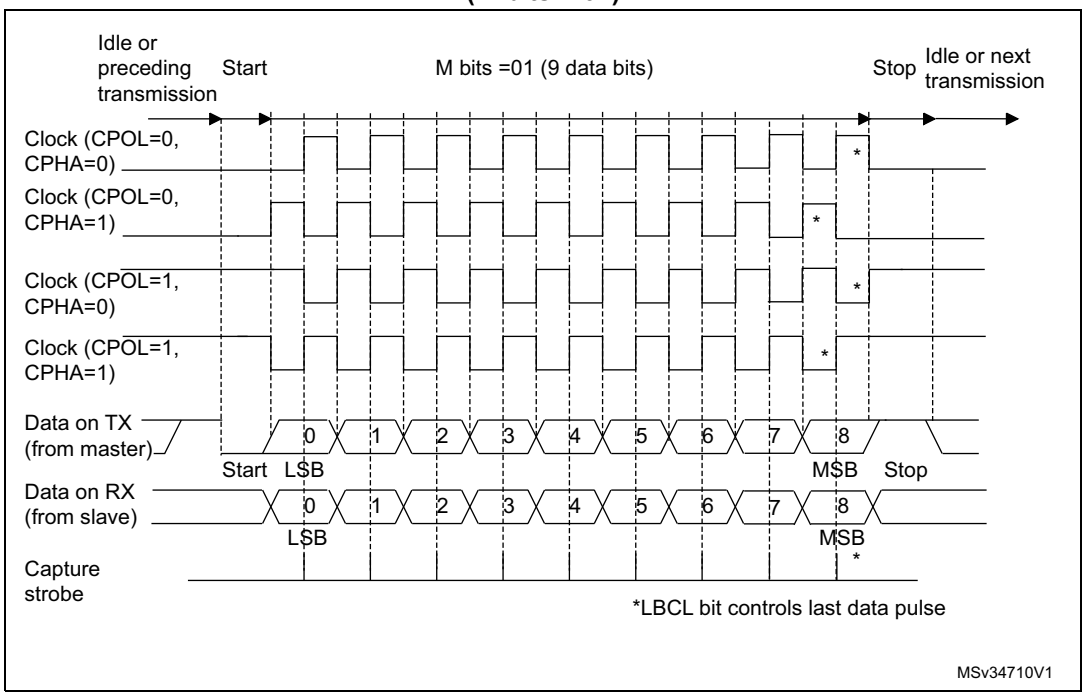


Figure 805. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

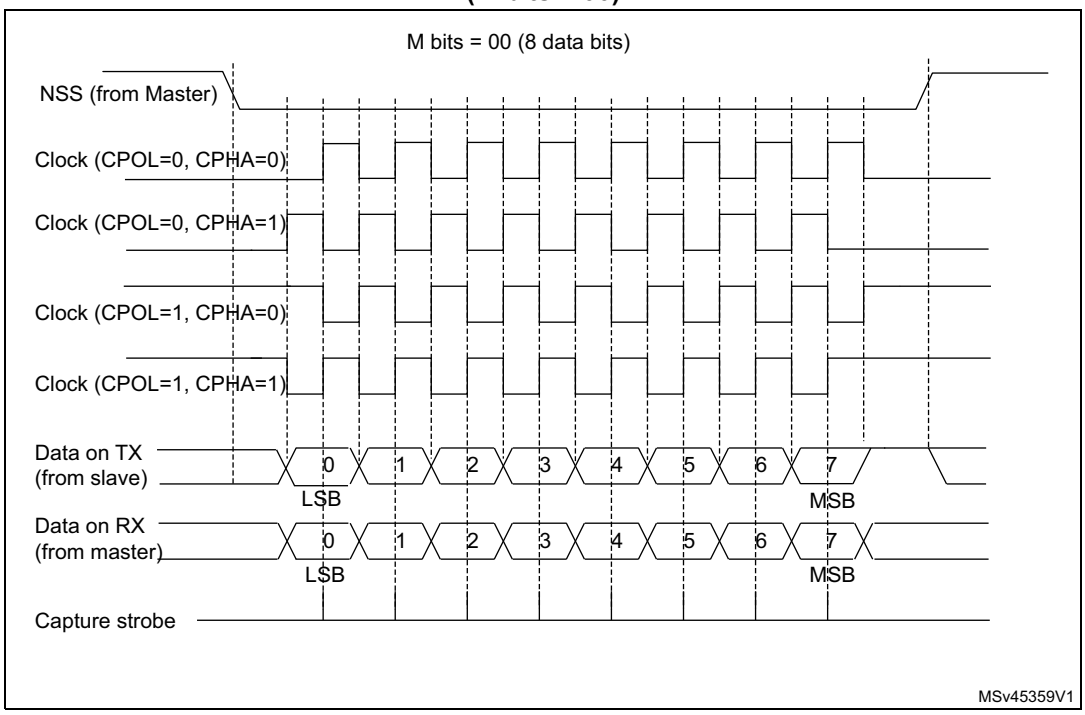
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 806*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 806. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

67.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

67.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

67.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 67.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

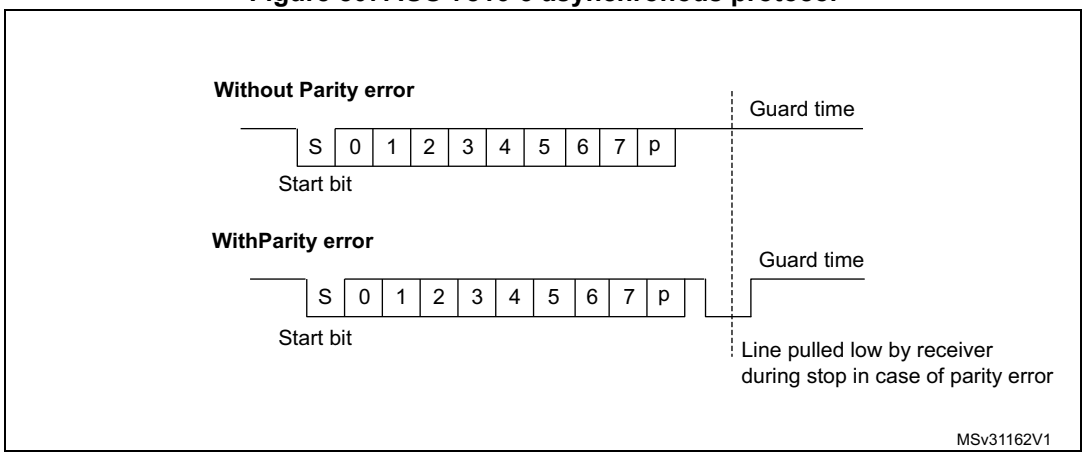
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 807 shows examples of what can be seen on the data line with and without parity error.

Figure 807. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

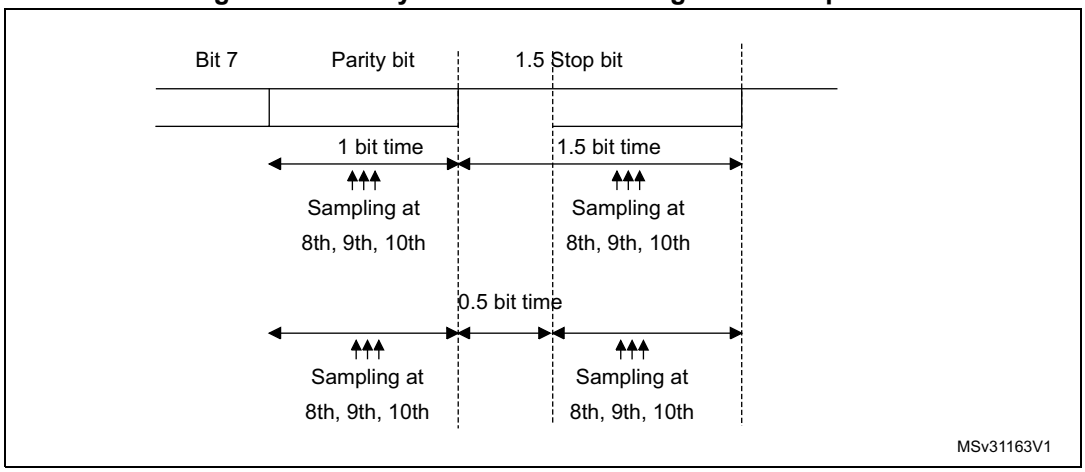
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 808 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 808. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

67.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 67.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 809*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 810*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 809. IrDA SIR ENDEC block diagram

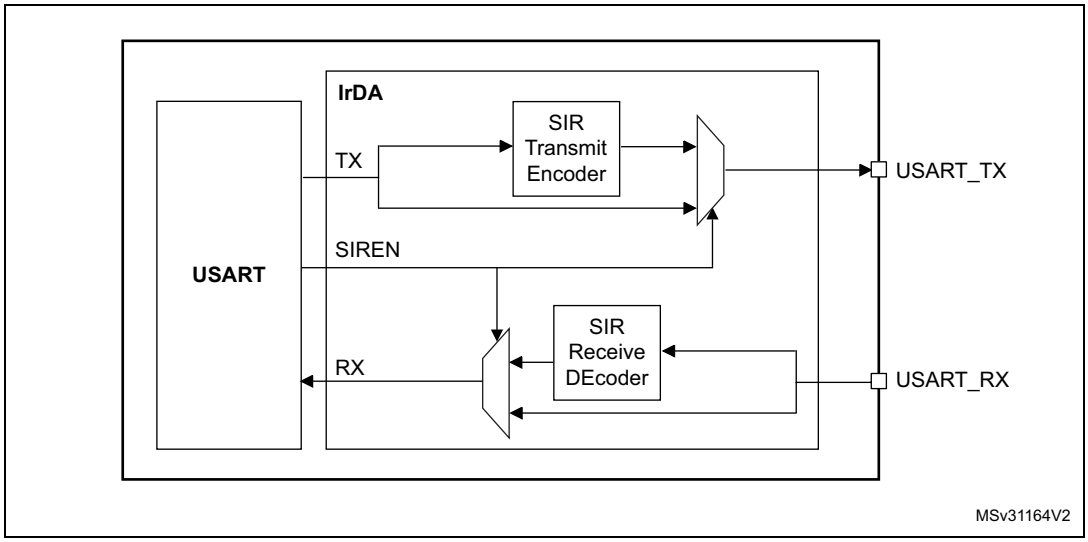
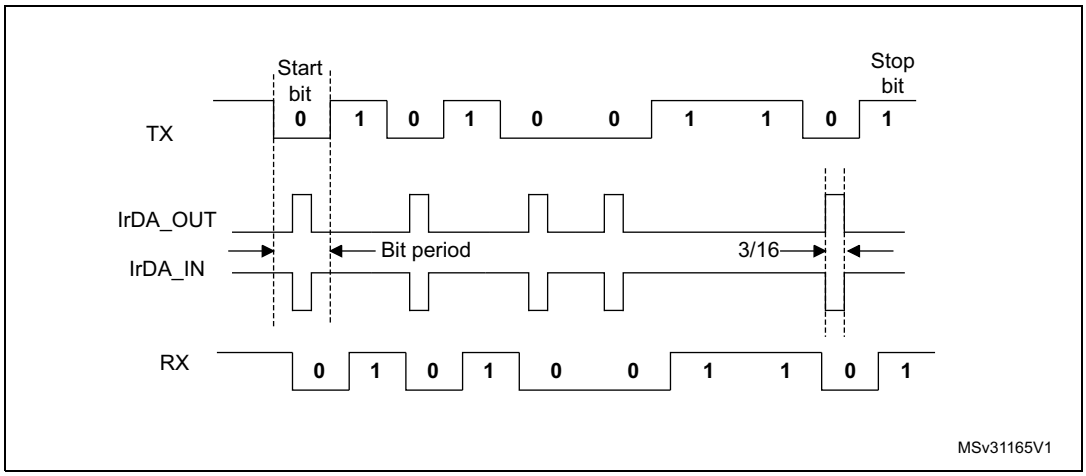


Figure 810. IrDA data modulation (3/16) - normal mode



67.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 67.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 67.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

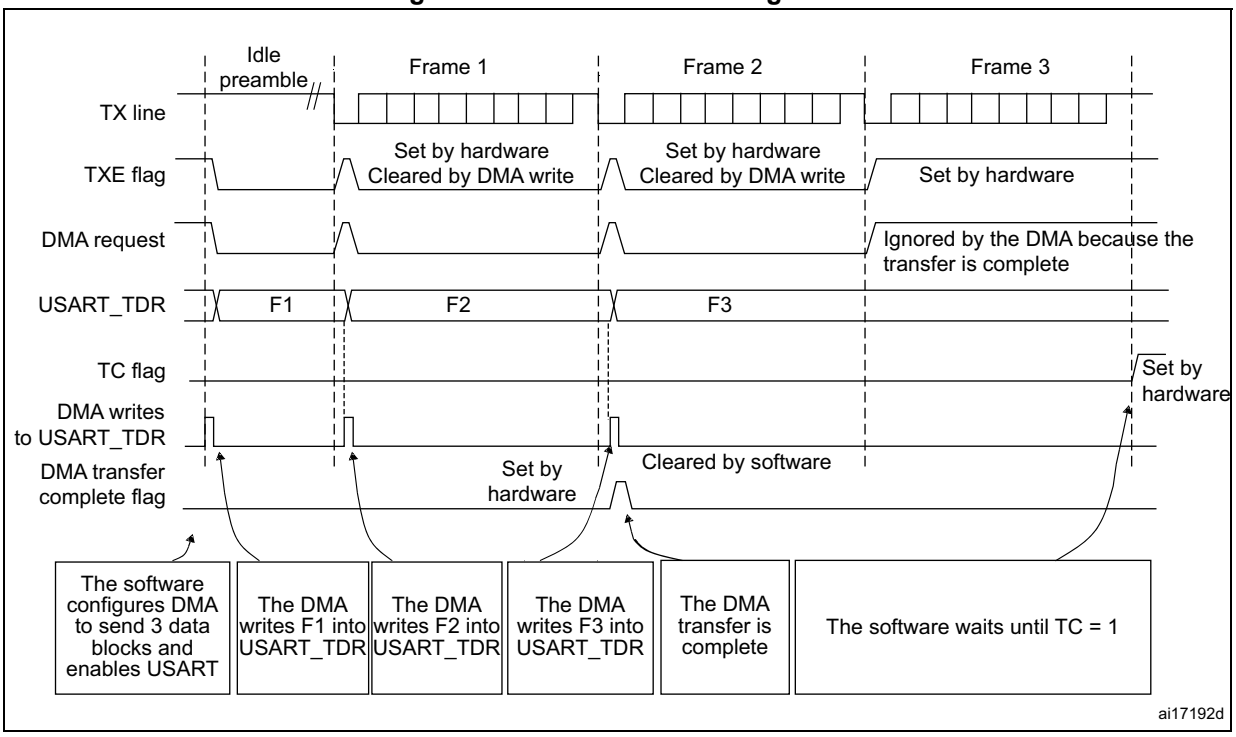
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 811. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

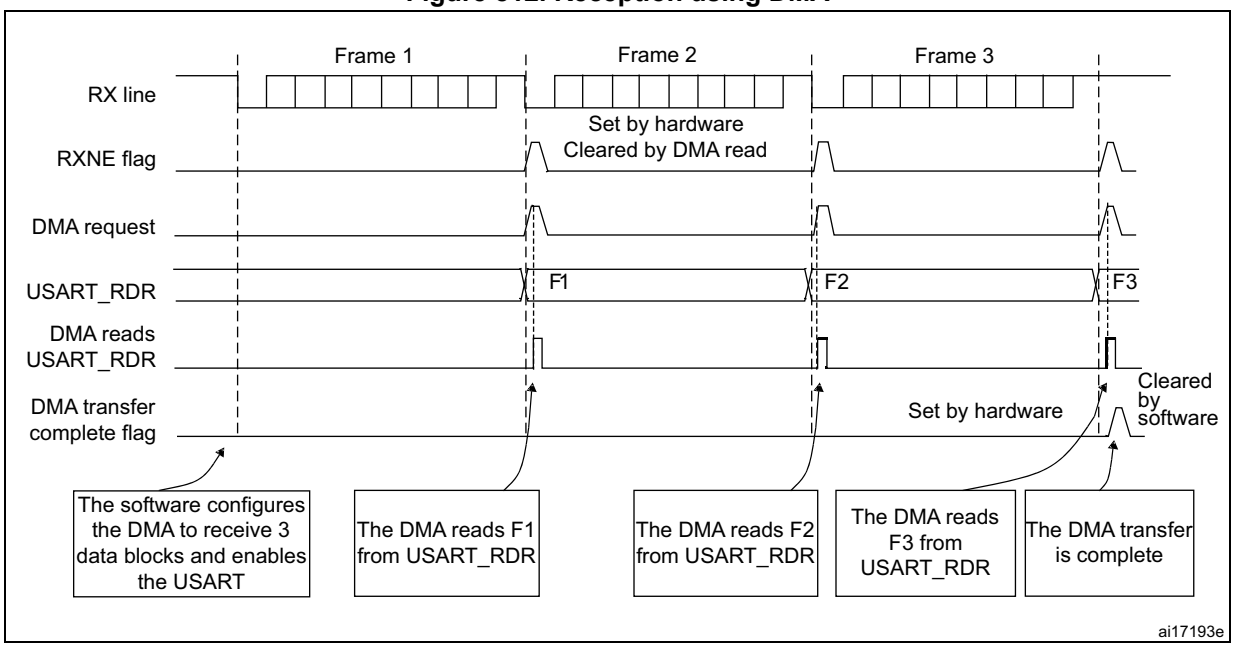
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 812. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

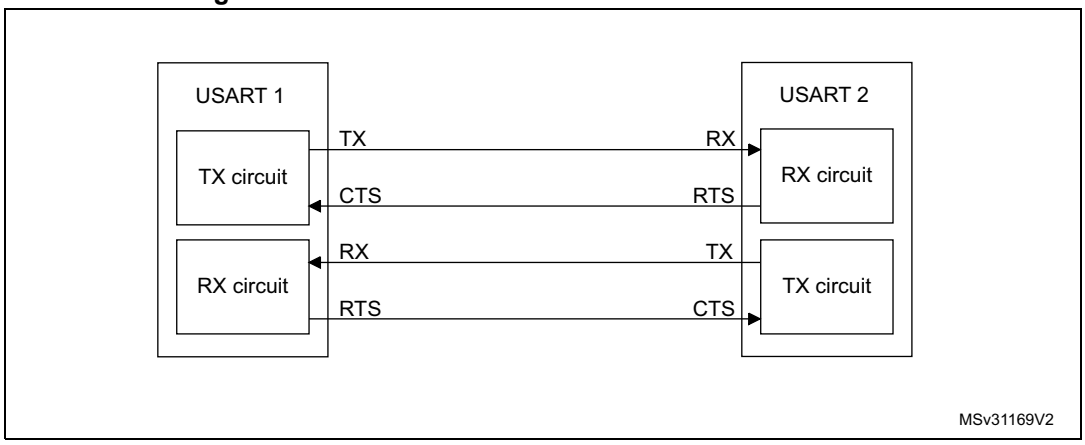
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

67.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 813* shows how to connect two devices in this mode:

Figure 813. Hardware flow control between two USARTs

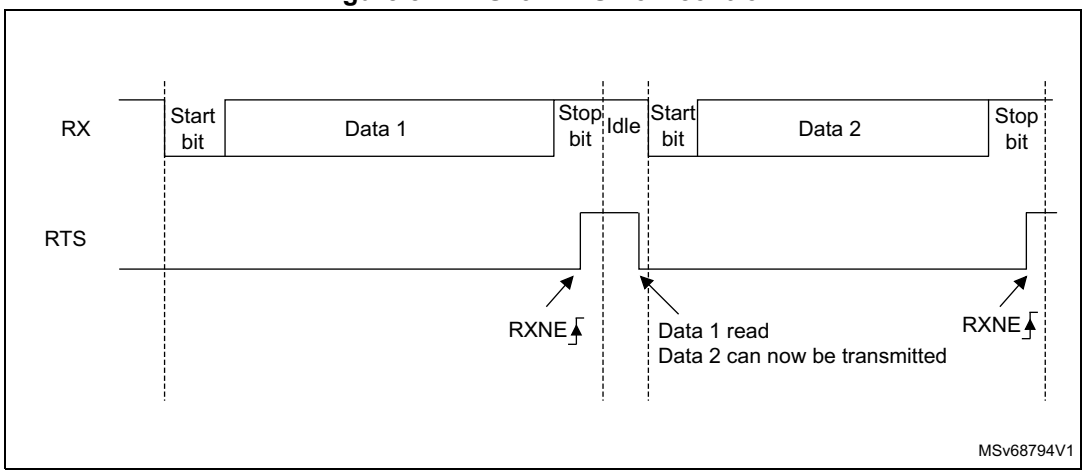


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 814* shows an example of communication with RTS flow control enabled.

Figure 814. RS232 RTS flow control



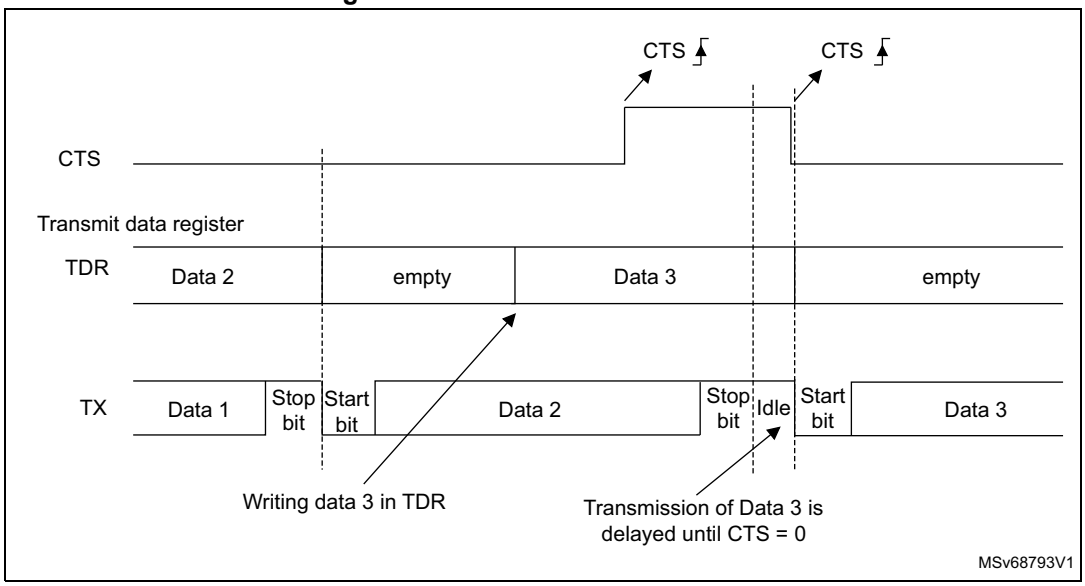
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 815* shows an example of communication with CTS flow control enabled.

Figure 815. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

67.5.22 USART autonomous mode

The USART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the USART status needs to be updated. Once the USART receives the APB clock, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the kernel and APB clocks are still available for the USART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

USART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the USART_AUTOCR register. It sets the TE bit in the USART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the USART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

USART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1 or DMAR = 1)
- If the FIFO mode is disabled, the APB clock is requested when the USART finishes sampling data and it is ready to be written in the USART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation of an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

The APB clock is also requested in reception mode when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Only UART and SPI master modes support the autonomous mode.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 67.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 686: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

67.6 USART in low-power modes

Table 689. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. If the USART is clocked by an oscillator available in Stop mode, transfers in asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 67.4* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

67.7 USART interrupts

Refer to *Table 690* for a detailed description of all USART interrupt requests.

Table 690. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 690. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(4)	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(4)	
	Overrun error in multibuffer communication.	ORE(5)		Write 1 in ORECF		Yes	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		Yes(4)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(6)	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 67.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
5. When OVRDIS = 0.
6. The DMA must be used when the FIFO mode is enabled.

67.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

67.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 67.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

67.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 67.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

67.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 797* and *Figure 798*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

67.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	Res.	Res.	Res.	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 67.4: USART implementation on page 3611.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

67.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCARCNT[2:0]			Res.
							rw					rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 67.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

67.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

67.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 67.4: USART implementation.

67.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 67.4: USART implementation.

67.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

67.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 67.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by writing to the USART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 67.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

67.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	Res.	RWU	SBKF	CMF	BUSY
						r			r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 67.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 67.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

67.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

67.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 67.5.1: USART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

67.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 67.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

67.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 67.4: USART implementation.

67.8.16 USART autonomous mode control register (USART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description of USART interconnections*.
This bitfield can be written only when the UE bit is cleared in USART_CR1 register.
0000: usart_trg0 selected
0001: usart_trg1 selected
...
1111: usart_trg15 selected

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in USART_CR1)
1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit
0: Trigger disabled
1: Trigger enabled
Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.
When a trigger is detected, TE is set to 1 in USART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit
This bitfield can be written only when the UE bit is cleared in USART_CR1 register.
0: Trigger active on rising edge
1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDN transmission data number
This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in USART_CR1.

67.8.17 USART hardware configuration register 2 (USART_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0116 (USARTx)

Reset value: 0x0000 0106 (UARTx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **CFG3[3:0]**: USART hardware configuration 3
CFG3[3:0] = Autonomous mode

Bits 7:4 **CFG2[3:0]**: USART hardware configuration 2
CFG2[3:0] = SPI_SLAVE

Bits 3:0 **CFG1[3:0]**: USART hardware configuration 1
CFG1[3:0] = LOG_FIFO_DEPTH_RX
RXFIFO size = $2^{LOG_FIFO_DEPTH_RX}$

67.8.18 USART hardware configuration register 1 (USART_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x6011 1111 (USARTx)

Reset value: 0x6011 0110 (UARTx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CFG8[3:0]**: USART hardware configuration 8
CFG8[3:0] = LOG_FIFO_DEPTH_TX
TXFIFO size = $2^{LOG_FIFO_DEPTH_TX}$

Bits 27:24 **CFG7[3:0]**: USART hardware configuration 7
CFG7[3:0] = BAUDGEN_TYPE

Bits 23:20 **CFG6[3:0]**: USART hardware configuration 6
CFG6[3:0] = DUAL_clock

Bits 19:16 **CFG5[3:0]**: USART hardware configuration 5
CFG5[3:0] = ABR_RTO_SCT1

Bits 15:12 **CFG4[3:0]**: USART hardware configuration 4
CFG4[3:0] = SMARTCARD_SEL

Bits 11:8 **CFG3[3:0]**: USART hardware configuration 3
CFG3[3:0] = IRDA_SEL

Bits 7:4 **CFG2[3:0]**: USART hardware configuration 2
CFG2[3:0] = LIN_MASTER

Bits 3:0 **CFG1[3:0]**: USART hardware configuration 1
CFG1[3:0] = SCLK_EXTRACT

67.8.19 USART version register (USART_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0034

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: USART major revision

Bits 3:0 **MINREV[3:0]**: USART minor revision

67.8.20 USART identification register (USART_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** USART identifier
ID[31:0] = 0x0013 0003

67.8.21 USART size identification register (USART_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification
These bits return the size of the memory region allocated to USART registers.
The size of this memory region is of 1 Kbytes.
SID[31:0] = 0xA3C5 DD01

67.8.22 USART register map

Table 691. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0					0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 691. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF		CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0
0x30	USART_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]															
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	USART_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value (USARTx)	0x0000 0116																															
	Reset value (UARTx)	0x0000 0106																															
0x3F0	USART_HWCFGR1	CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value (USARTx)	0x6011 1111																															
	Reset value (UARTx)	0x6011 0110																															
0x3F4	USART_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]		
	Reset value	0x0000 0034																															
0x3F8	USART_IPIDR	ID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3FC	USART_SIDR	SID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

68 Low-power universal asynchronous receiver transmitter (LPUART)

68.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

68.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

68.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 692. Instance implementation on STM32MP21xx

Instance	STM32MP21xx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
UART4	Basic
UART5	Basic
UART7	Basic
LPUART1	Low-power

Table 693. USART/LPUART features

Modes/features(1)	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X(2)
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-

Table 693. USART/LPUART features (continued)

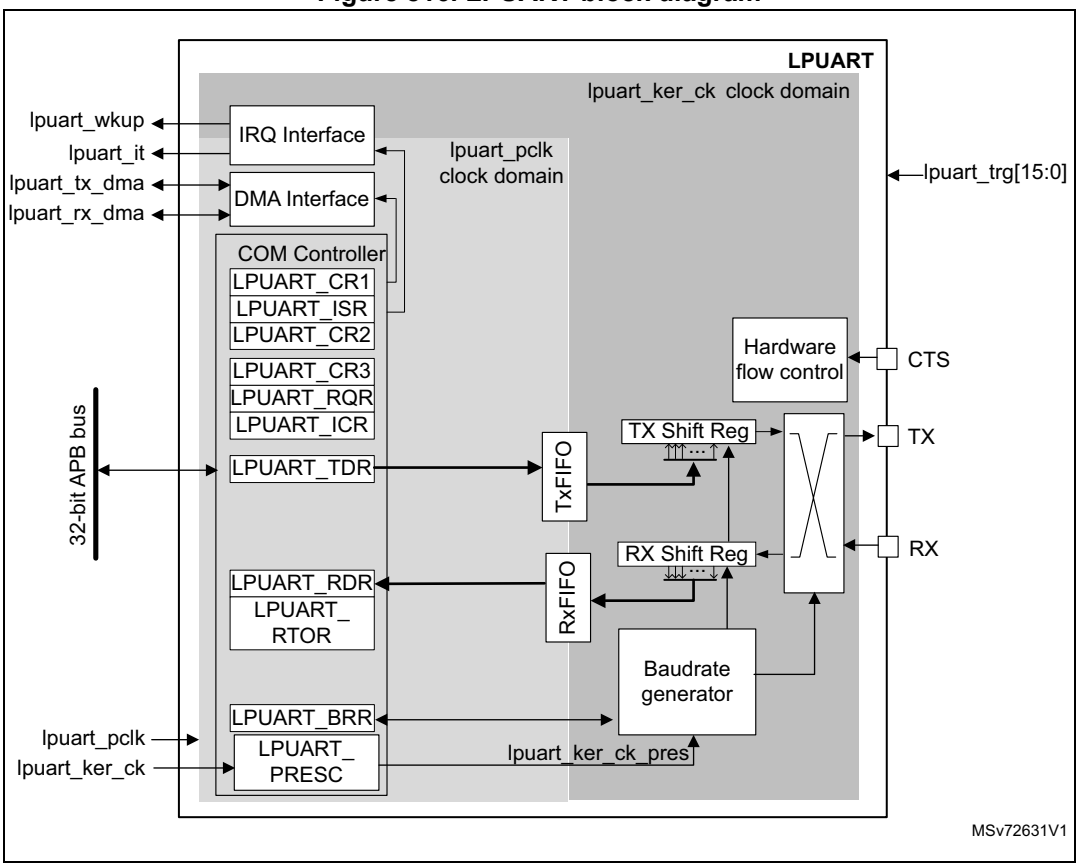
Modes/features[1]	Full feature	Basic feature	Low-power feature
Dual-clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	-
USART data length	7, 8, and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	64		8
Wake-up from low-power mode	X[3]	X[3]	X[3]
Autonomous mode	X	X	X

1. X = supported.
2. Only the RS232 CTS flow control is supported.
3. Wake-up supported from Stop 1, Stop 2, Low-power stop 1, and Low-power stop 2 modes.

68.4 LPUART functional description

68.4.1 LPUART block diagram

Figure 816. LPUART block diagram



68.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pin:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

Refer to *Table 694* and *Table 695* for the list of LPUART input/output pins and internal signals.

Table 694. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send

Description LPUART input/output signals

Table 695. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request
lpuart_trg[15:0]	Input	LPUART triggers

Description LPUART interconnections

Table 696. LPUART interconnections (LPUART1)

Signal name	Source
lpuart_trg0	-
lpuart_trg1	-
lpuart_trg2	-
lpuart_trg3	-
lpuart_trg4	exti4 (from EXIT2)
lpuart_trg5	exti5 (from EXTI2)
lpuart_trg6	-
lpuart_trg7	lptim3_ch1
lpuart_trg8	lptim4_ch1
lpuart_trg9	lptim5_out
lpuart_trg10	rtc_alra_trg
lpuart_trg11	rtc_wut_trg
lpuart_trg12	exti6 (from EXTI1)
lpuart_trg13	exti7 (from EXTI1)
lpuart_trg14	lptim1_ch1
lpuart_trg15	lptim2_ch1

68.4.3 LPUART clocks

The simplified block diagram given in *Section 68.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

68.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 792*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

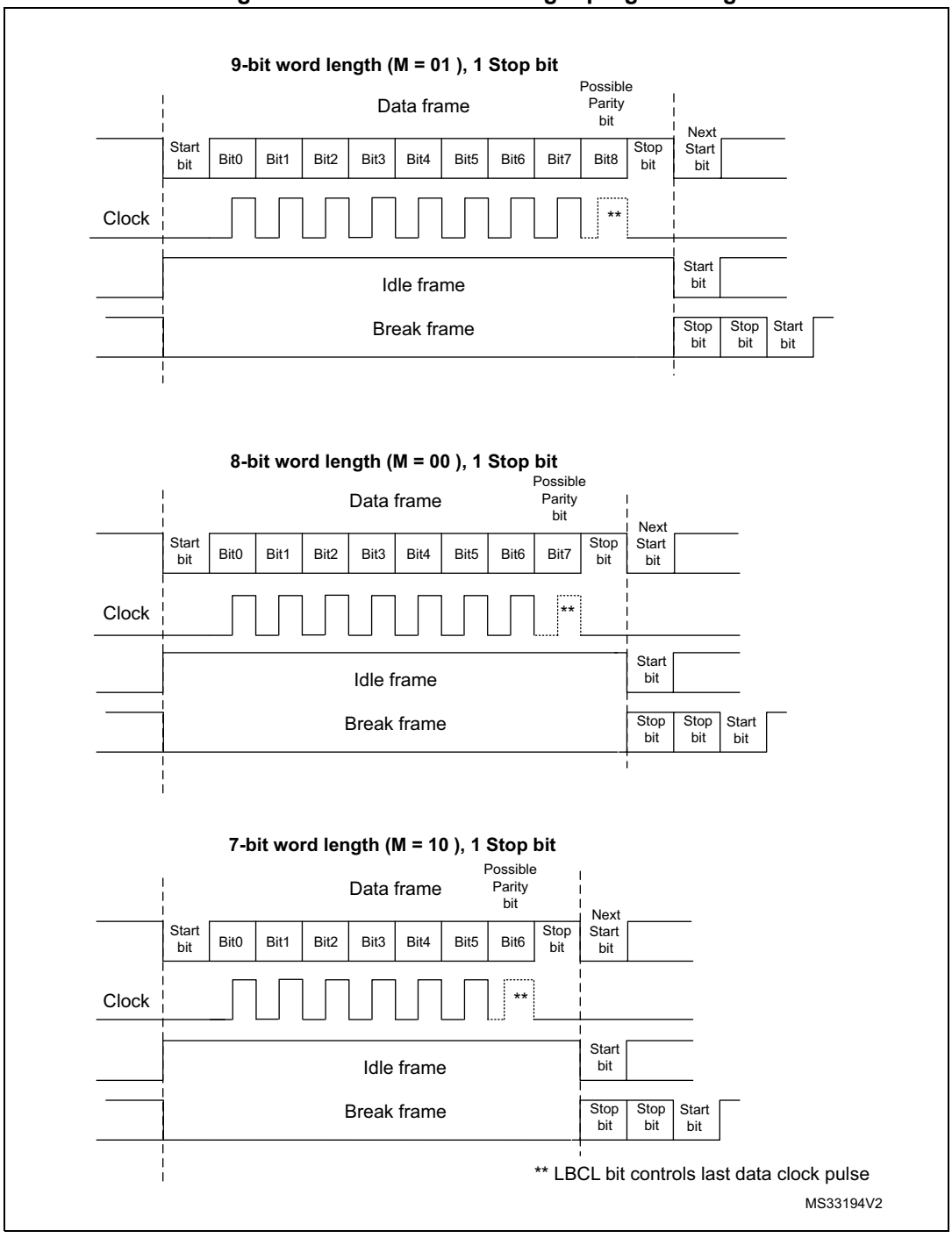
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 817. LPUART word length programming



68.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

68.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 68.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

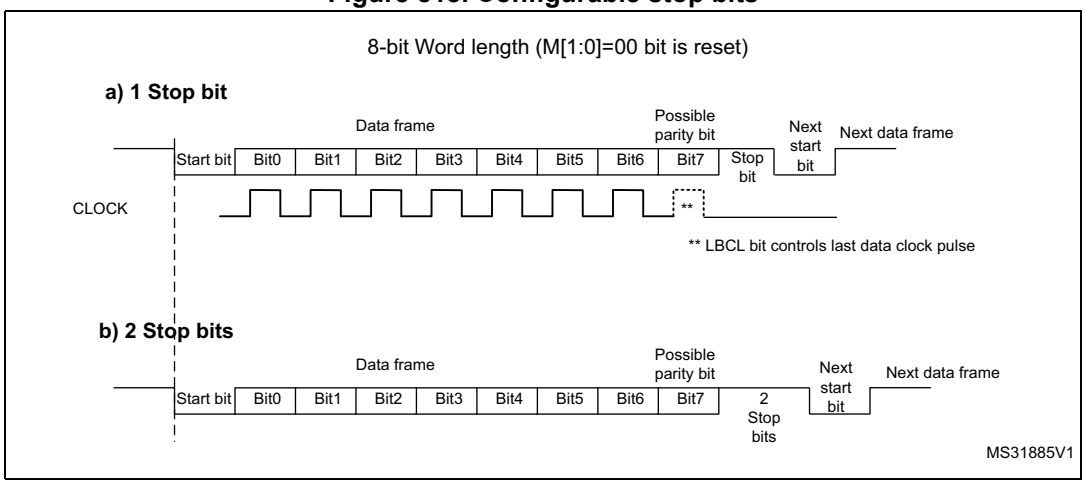
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 818. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 68.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

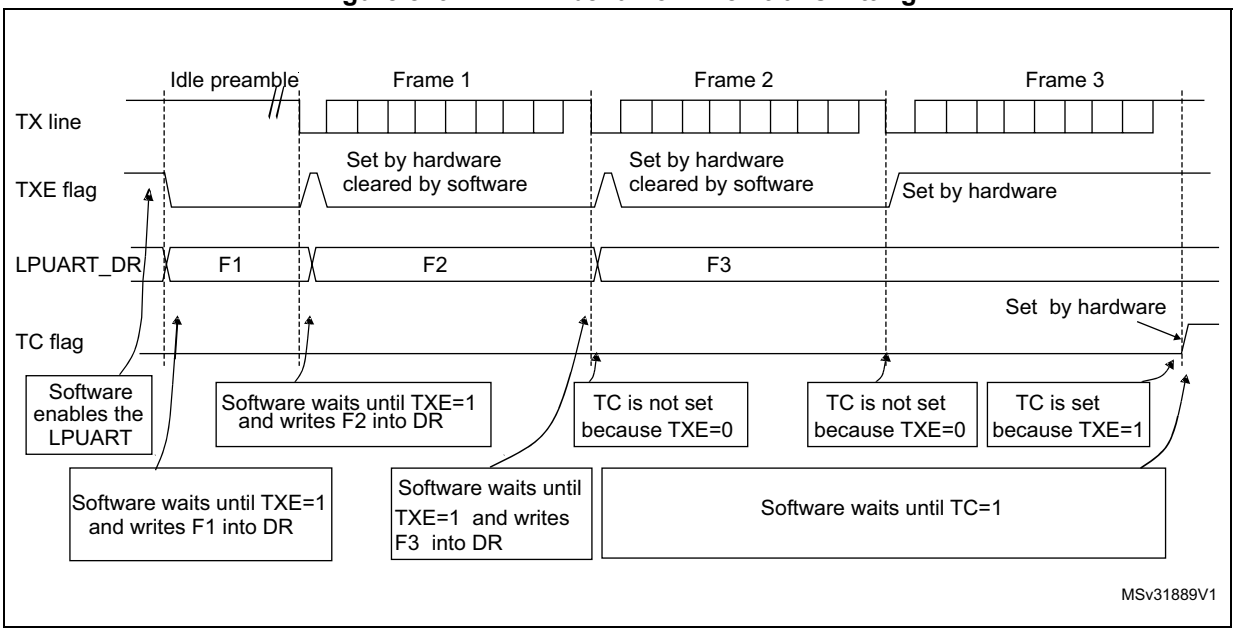
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 819: TC/TXE behavior when transmitting*).

Figure 819. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 817*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

68.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 68.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

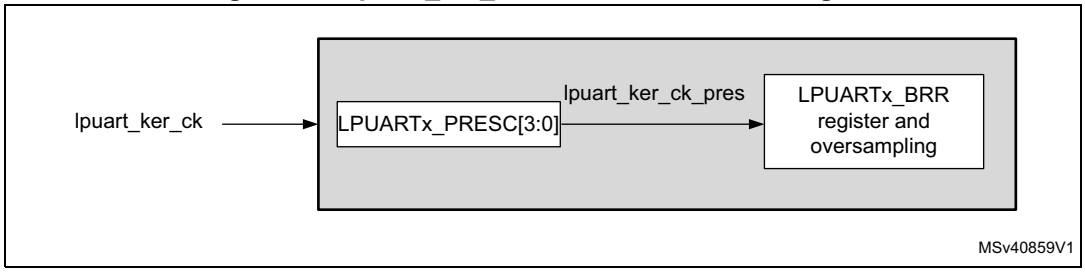
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 820. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

68.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 697. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

68.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 698*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 698. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 698 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

68.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

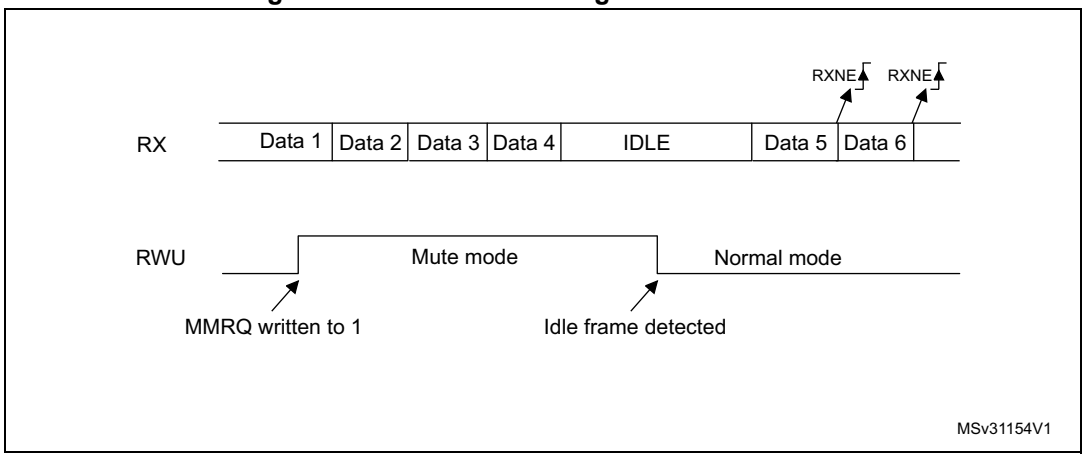
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 821*.

Figure 821. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

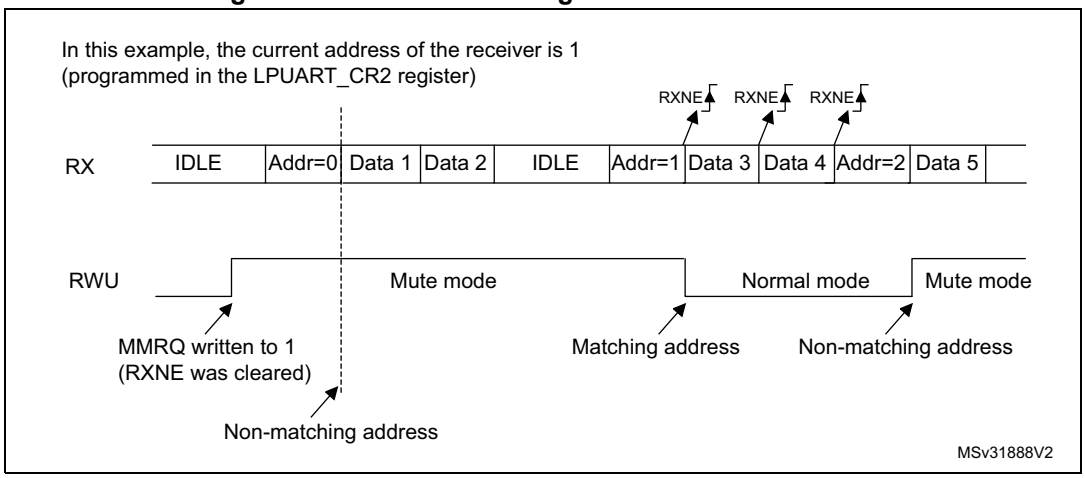
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 822*.

Figure 822. Mute mode using address mark detection



68.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 699*.

Table 699: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

68.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

68.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 68.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 68.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

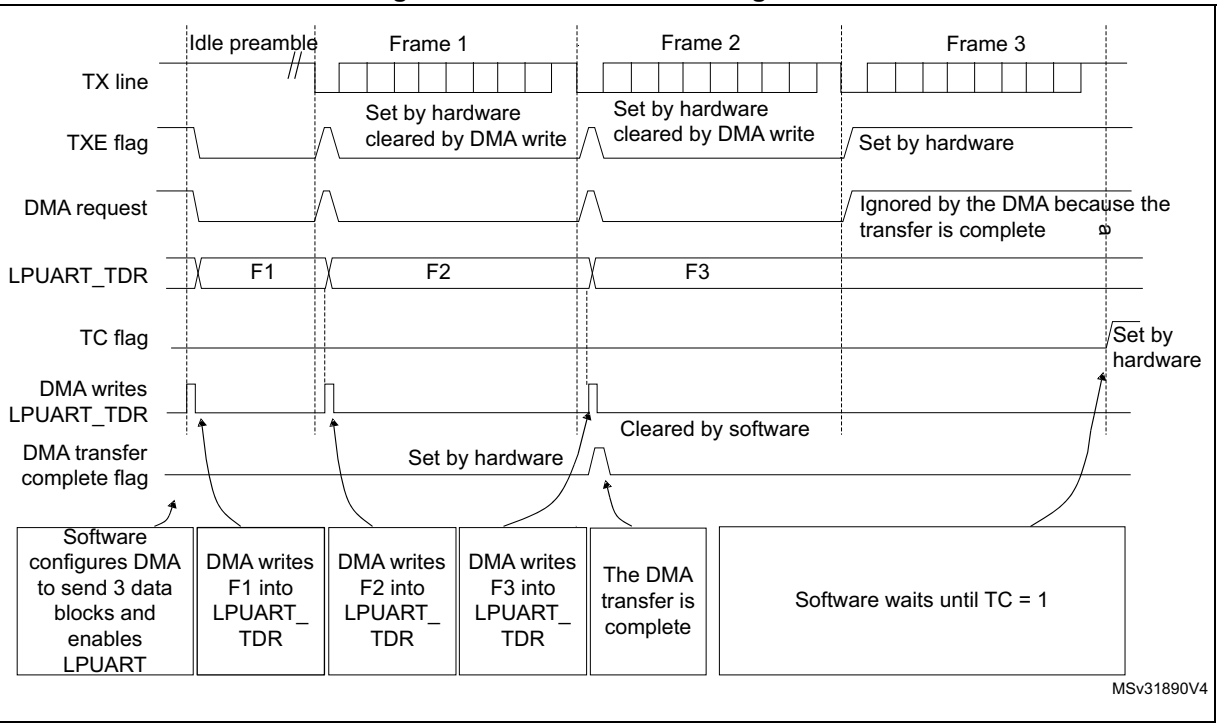
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 823. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

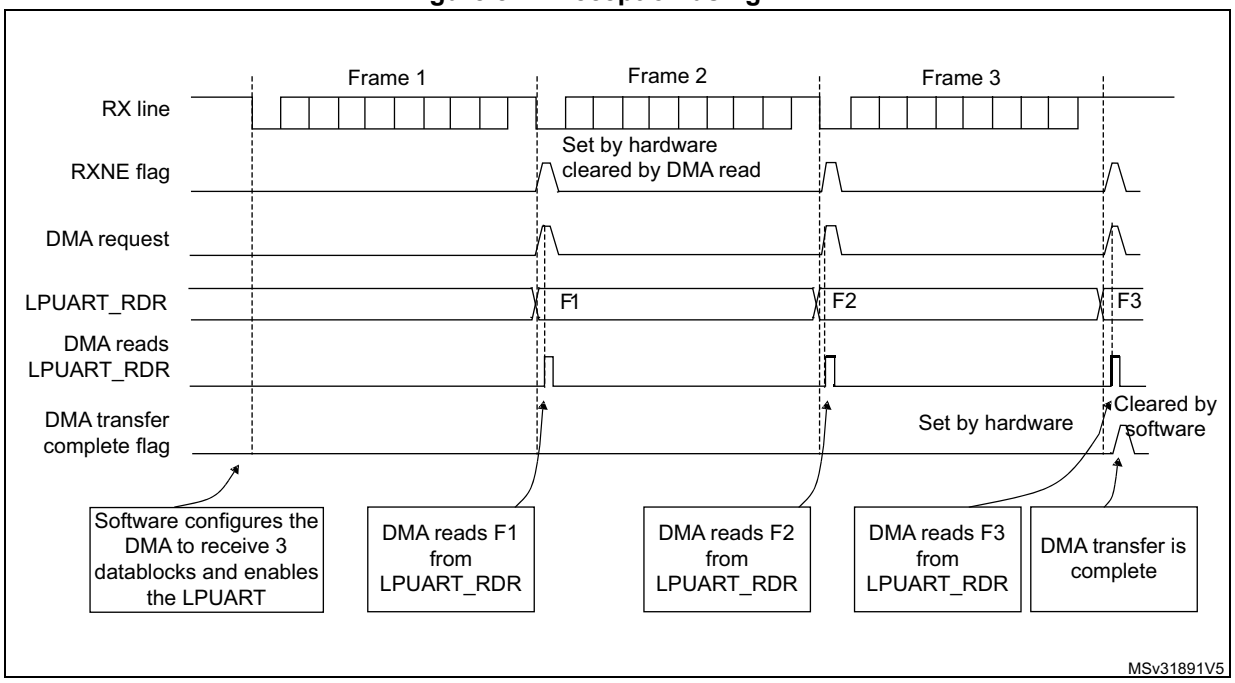
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 824. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

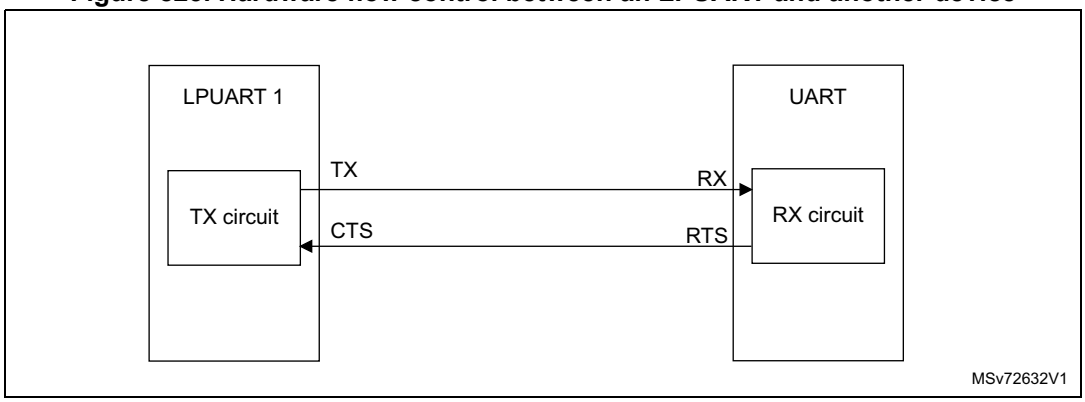
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

68.4.14 RS232 hardware flow control

It is possible to control the serial data flow between an LPUART and another device, by using the LPUART CTS input and the other device RTS output. *Figure 825* shows the connection in this mode.

Figure 825. Hardware flow control between an LPUART and another device



The RS232 CTS flow control can be enabled by writing the CTSE bit of the LPUART_CR3 register to 1.

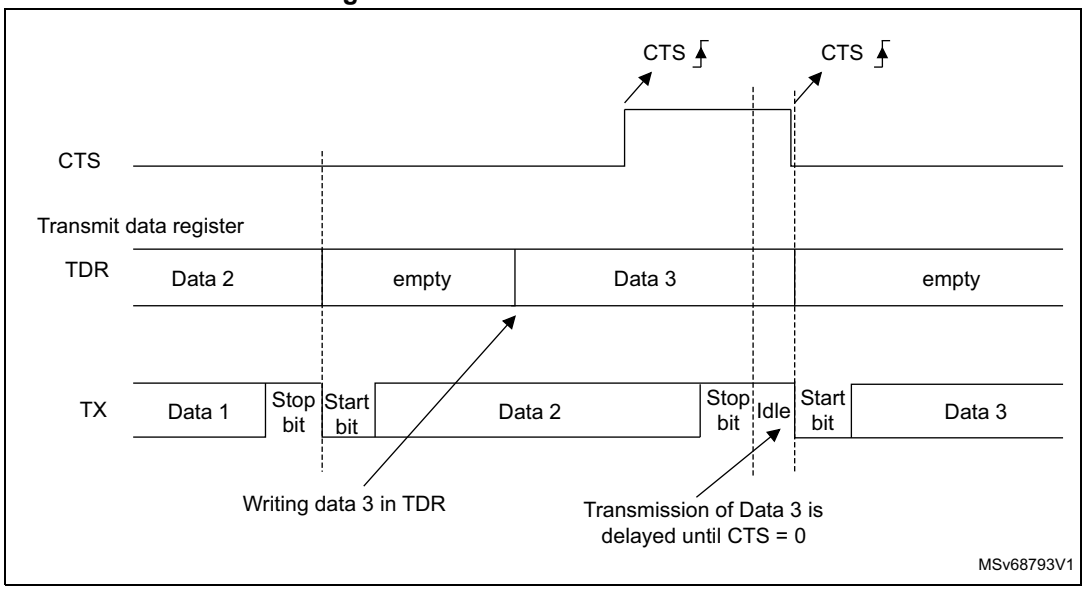
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 826* shows an example of communication with CTS flow control enabled.

Figure 826. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

68.4.15 LPUART autonomous mode

The LPUART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the LPUART status needs to be updated. Once the LPUART receives the kernel and APB clocks, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the APB clock is still available for the LPUART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

LPUART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the LPUART_AUTOCR register. It sets the TE bit in the LPUART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the LPUART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

LPUART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1)
- If the FIFO mode is disabled, the APB clock is requested when the LPUART finishes sampling data and it is ready to be written in the LPUART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

In reception mode, the APB clock is requested when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 68.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 698: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

68.5 LPUART in low-power modes

Table 700. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. If the LPUART is clocked by an oscillator available in Stop mode, transfers in asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 68.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

68.6 LPUART interrupts

Refer to *Table 701* for a detailed description of all LPUART interrupt requests.

Table 701. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	.Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(3)	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(3)	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		Yes	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		Yes(3)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(5)	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 68.3: LPUART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
4. When OVRDIS = 0.
5. The DMA must be used when the FIFO mode is enabled.

68.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

68.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 67.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 68.4.14: RS232 hardware flow control*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not functional in low-power mode.
1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.
If the LPUART *does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 68.3: LPUART implementation.*

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

68.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 67.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 68.4.14: RS232 hardware flow control*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is disabled, TE bit is set and cleared by software. When the autonomous mode is enabled, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 68.3: LPUART implementation.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

68.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

68.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	Res.	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw		rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 Reserved, must be kept at reset value.

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

68.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	Res.	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw		rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 8 Reserved, must be kept at reset value.

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

68.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

68.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

68.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by writing to the LPUART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

68.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
									r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

68.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

68.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 68.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

68.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 68.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

68.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 68.3: LPUART implementation.

68.7.14 LPUART autonomous mode control register (LPUART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description LPUART interconnections*.

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0000: lpuart_trg0 selected

0001: lpuart_trg1 selected

...

1111: lpuart_trg15 selected

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in LPUART_CR1)

1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in LPUART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDC transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in LPUART_CR1.

68.7.15 LPUART hardware configuration register 2 (LPUART_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0103

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **CFG3[3:0]**: LPUART hardware configuration 3
CFG3[3:0]: Autonomous mode

Bits 7:4 **CFG2[3:0]**: LPUART hardware configuration 2
CFG2[3:0]: SPI_SLAVE

Bits 3:0 **CFG1[3:0]**: LPUART hardware configuration 1
CFG1[3:0]: LOG_FIFO_DEPTH_RX
RXFIFO size = $2^{LOG_FIFO_DEPTH_RX}$

68.7.16 LPUART hardware configuration register 1 (LPUART_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x3110 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CFG8[3:0]**: LPUART hardware configuration 8
CFG8[3:0] = LOG_FIFO_DEPTH_TX
TXFIFO size = $2^{LOG_FIFO_DEPTH_TX}$

Bits 27:24 **CFG7[3:0]**: LPUART hardware configuration 7
CFG7[3:0] = BAUDGEN_TYPE

Bits 23:20 **CFG6[3:0]**: LPUART hardware configuration 6
CFG6[3:0] = DUAL_clock

Bits 19:16 **CFG5[3:0]**: LPUART hardware configuration 5
CFG5[3:0] = ABR_RTO_SCT1

Bits 15:12 **CFG4[3:0]**: LPUART hardware configuration 4
CFG4[3:0] = SMARTCARD_SEL

Bits 11:8 **CFG3[3:0]**: LPUART hardware configuration 3
CFG3[3:0] = IRDA_SEL

Bits 7:4 **CFG2[3:0]**: LPUART hardware configuration 2
CFG2[3:0] = LIN_MASTER

Bits 3:0 **CFG1[3:0]**: LPUART hardware configuration 1
CFG1[3:0] = SCLK_EXTRACT

68.7.17 LPUART version register (LPUART_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0034

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits return the LPUART major revision.

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits return the LPUART minor revision.

68.7.18 LPUART identification register (LPUART_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** Peripheral identifier
These bits return the LPUART identifier ID[31:0].

68.7.19 LPUART size identification register (LPUART_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

These bits return the size of the memory region allocated to LPUART registers.

68.7.20 LPUART register map

Table 702. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1F IFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1F IFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	Res.	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0								0	0	0	0		0	0		0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	Res.	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value																	0	0	0	0		0	0		0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	

Table 702. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0
0x30	LPUART_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]			IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]																
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	LPUART_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[3:0]			CFG2[3:0]				CFG1[3:0]				
	Reset value	0x0000 0103																															
0x3F0	LPUART_HWCFGR1	CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]			CFG1[3:0]			
	Reset value	0x3110 0000																															
0x3F4	LPUART_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0034																															
0x3F8	LPUART_IPIDR	ID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3FC	LPUART_SIDR	SID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

69 Serial peripheral interface (SPI)

69.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

69.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

- Autonomous functionality in Stop modes (handling of the transfer flow and required clock distribution) with wake-up from Stop capability
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

69.3 SPI implementation

The table below describes the SPI implementation.

Table 703. SPI features

SPI feature	SPI1, SPI2, SPI3 (full-featured instances)	SPI4, SPI5, SPI6 (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits	Configurable from 4 to 16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length configurable from 5 to 17 bits
FIFO size	16x8 bits	8x8 bits
FIFO threshold	1 - 16 data	1 - 4 data
Number of data control (TSIZE)	Up to 65535	Up to 65535
I2S feature	Yes	No
Autonomous in Stop 1/2 modes with wake-up capability	Yes	Yes
Autonomous in LP-Stop 1/2 and Standby 1 modes, with wake-up capability	Yes	Yes

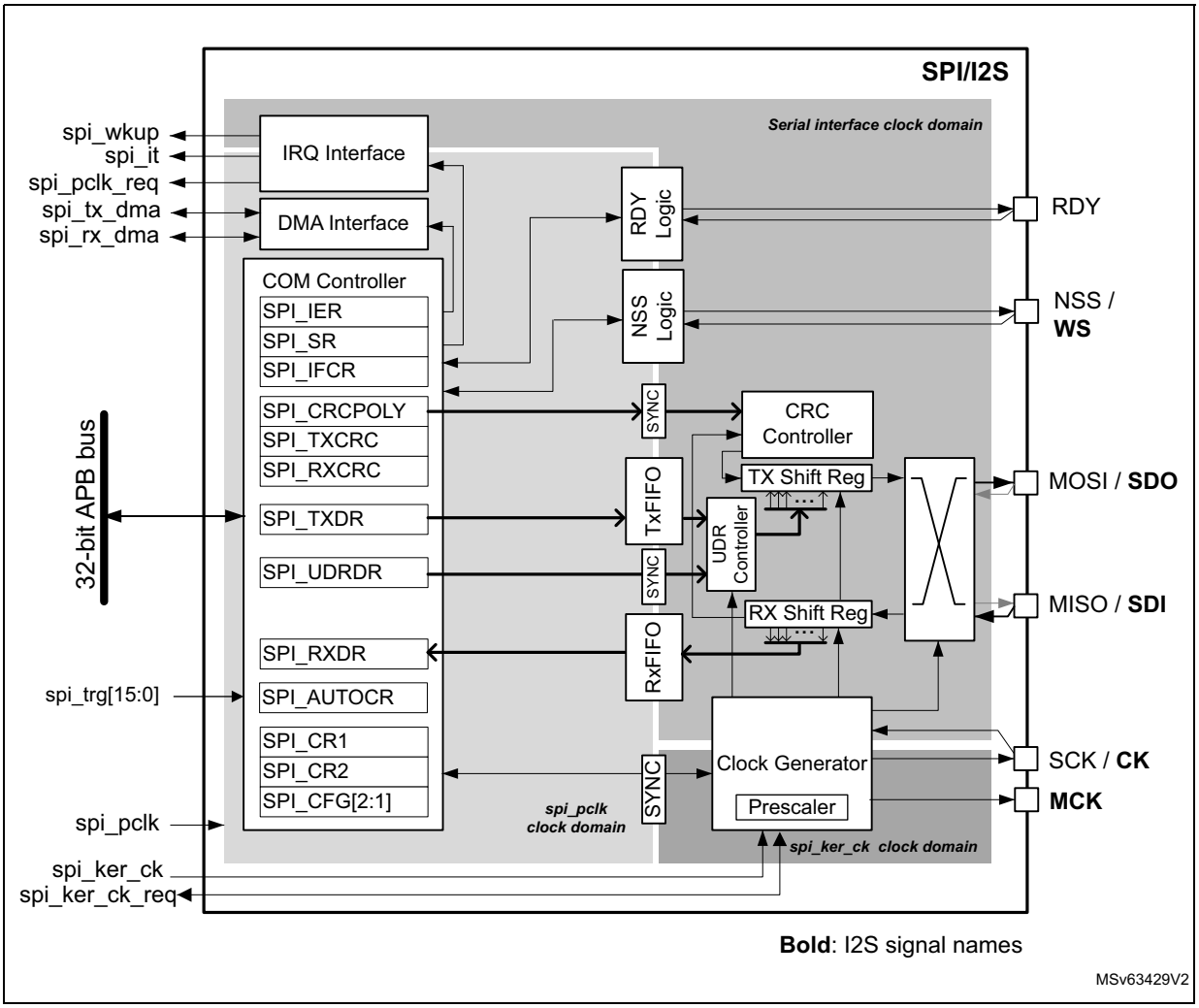
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 708: SPI wake-up and interrupt requests and Table 91: Functionalities depending on system operating mode.

69.4 SPI functional description

69.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 827*.

Figure 827. SPI/I2S block diagram



The simplified scheme of *Figure 827* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 69.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 69.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

69.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 69.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 69.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 704* and *Table 705* for the list of SPI input / output pins and internal signals.

Table 704. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input

Table 704. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 69.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 705. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request
spi_trg[15:0]	Input	SPI triggers

Description of SPI interconnections

Table 706. SPI interconnection (SPI1 to SPI6)

Signal name	Trigger source
spi_trg0	-
spi_trg1	-
spi_trg2	-
spi_trg3	-
spi_trg4	exti4 (EXTI2)
spi_trg5	exti5 (EXTI2)
spi_trg6	-
spi_trg7	lptim3_ch1
spi_trg8	lptim4_ch1
spi_trg9	lptim5_ch1
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg

Table 706. SPI interconnection (SPI1 to SPI6) (continued)

Signal name	Trigger source
spi_trg12	exti6 (EXTI1)
spi_trg13	exti7 (EXTI1)
spi_trg14	lptim1_ch1
spi_trg15	lptim2_ch1

69.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

69.4.4 Communications between one master and one slave

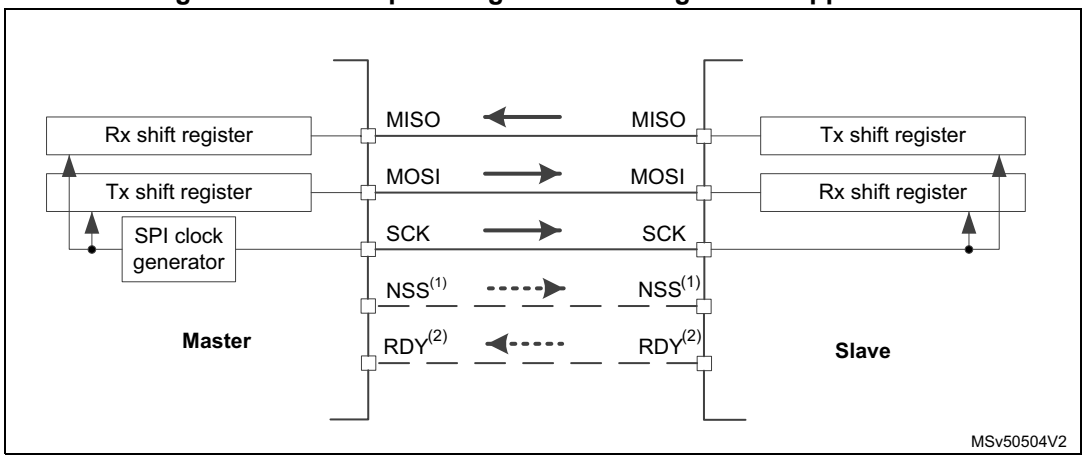
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 828. Full-duplex single master/ single slave application

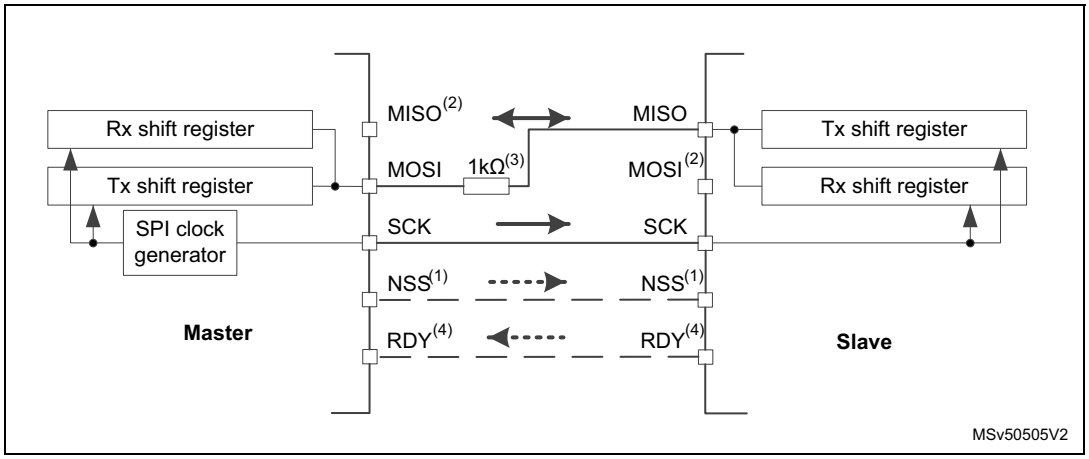


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 69.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 829. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 69.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between

MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.

4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 69.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

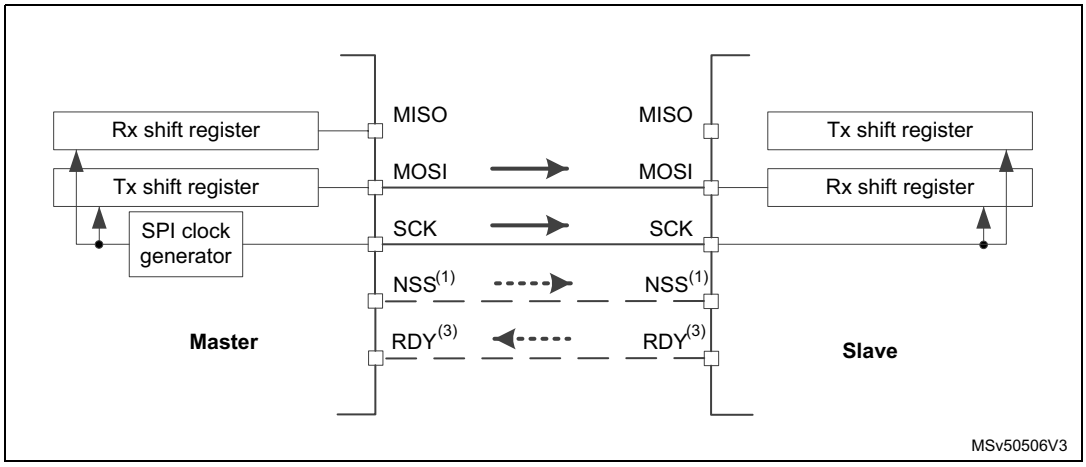
  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 69.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 830. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to

operate in a single master-slave pair (see *Section 69.4.7*).

2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

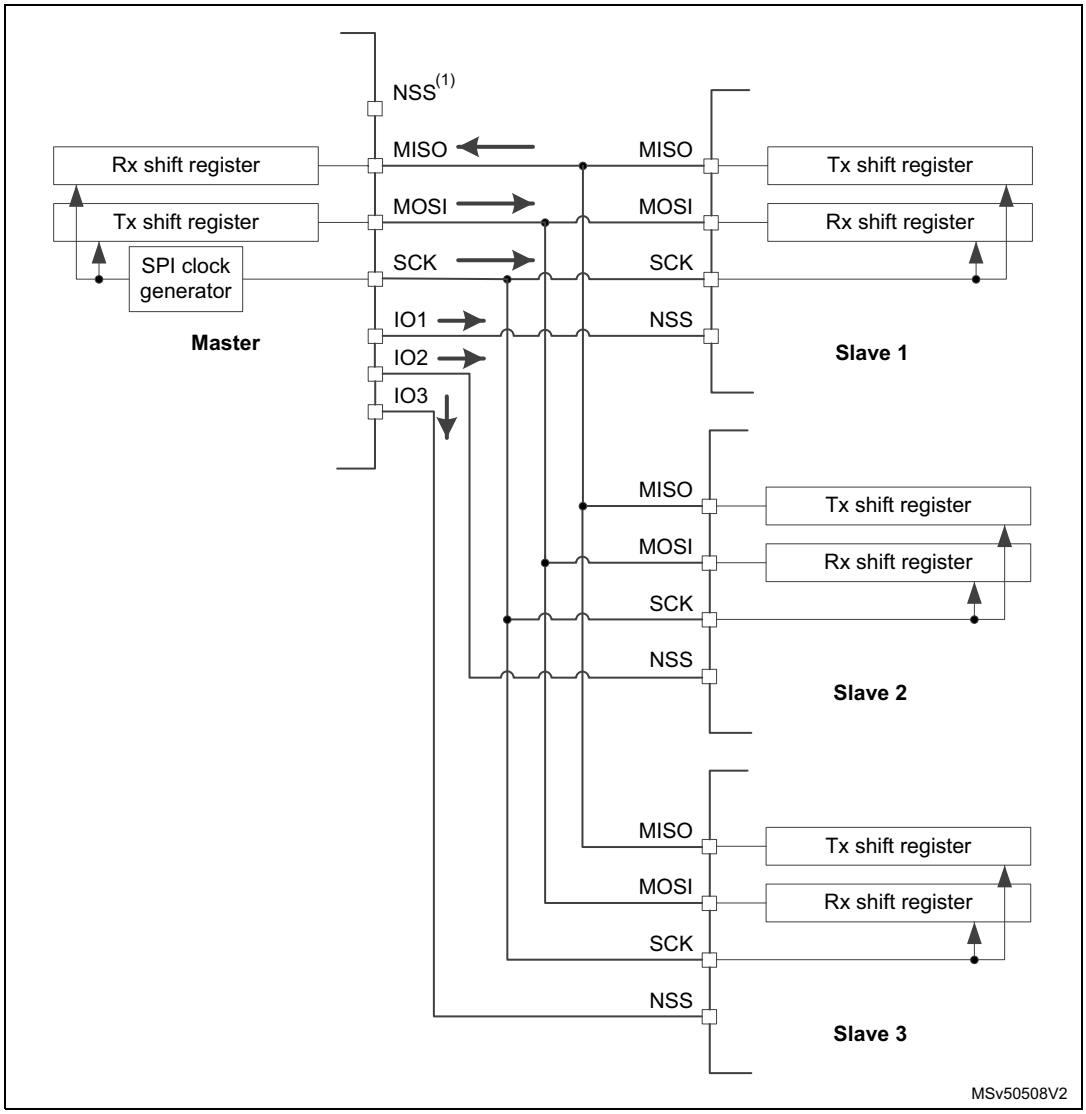
69.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 831.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 831. Master and three independent slaves connected in star topology



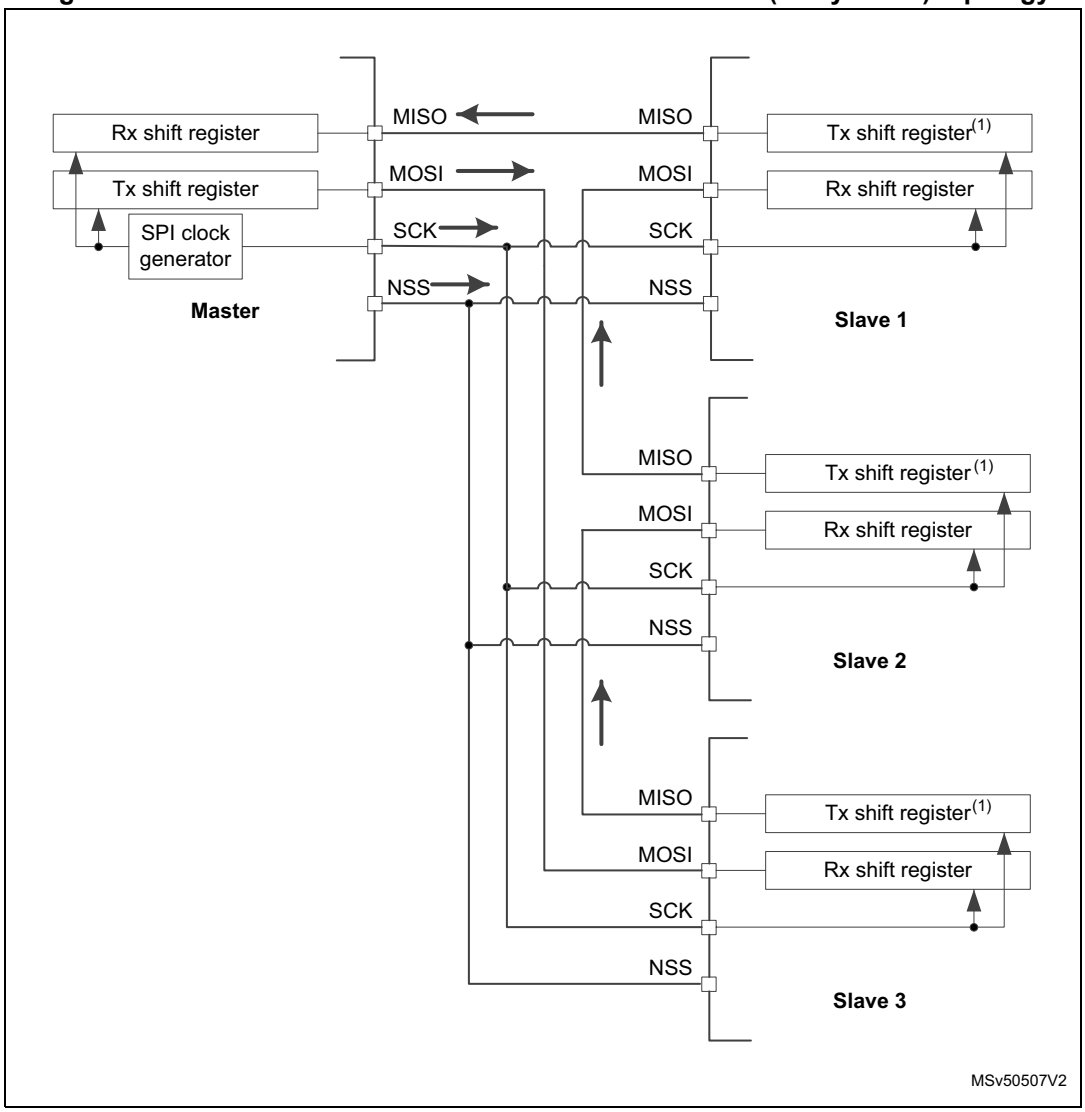
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 69.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 832*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 832. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

69.4.6 Multimaster communication

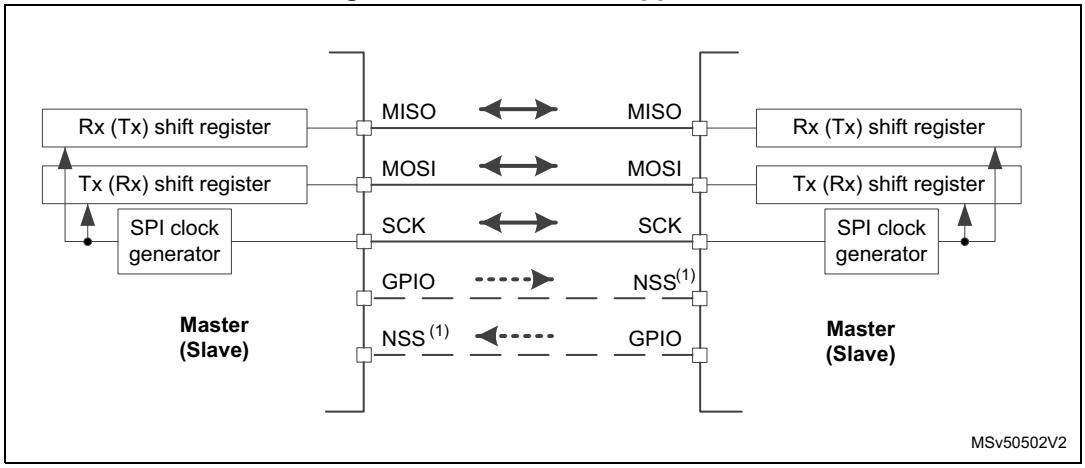
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 833. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

69.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

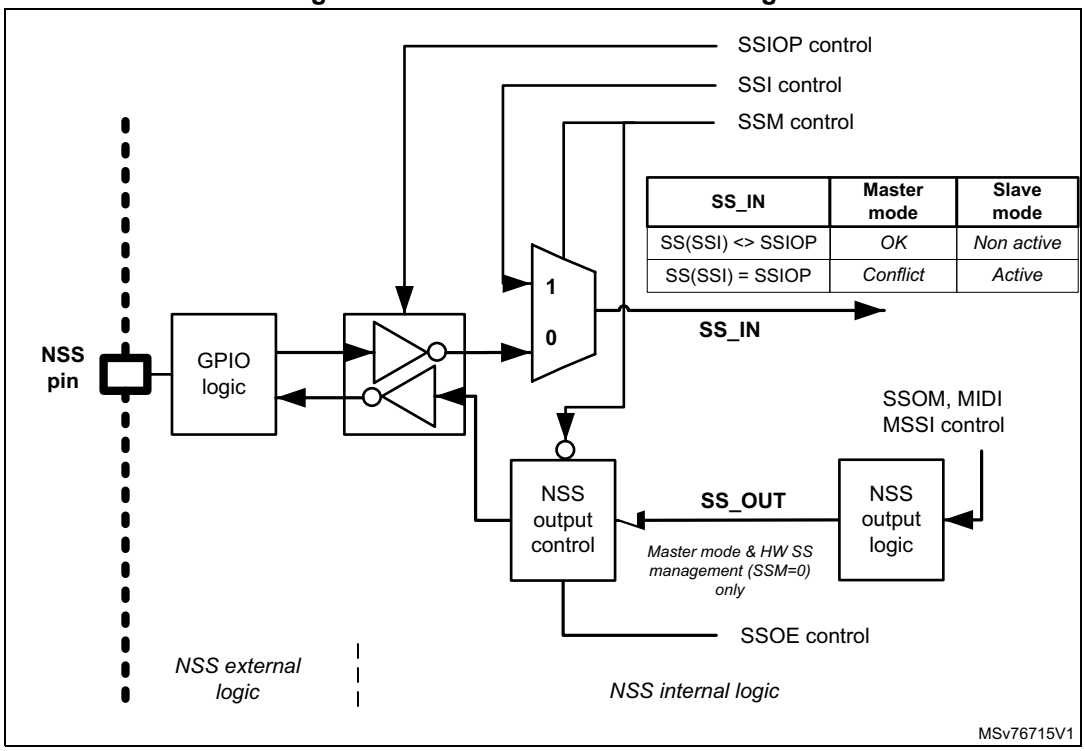
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

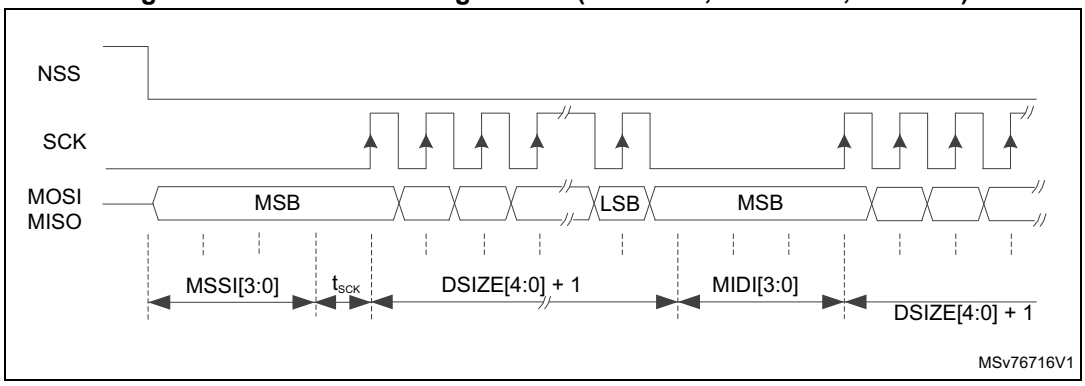
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 834. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 835*).

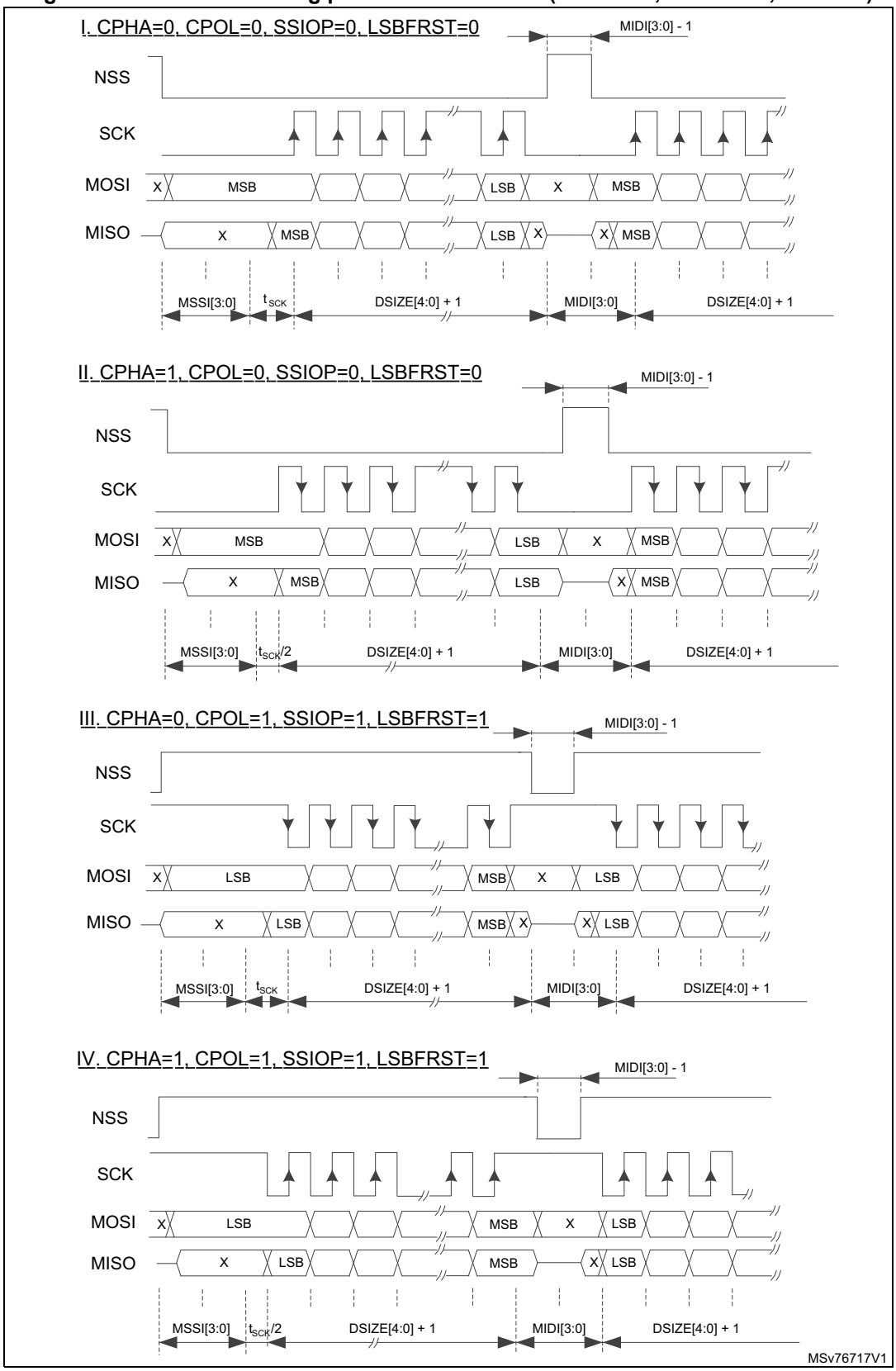
Figure 835. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 836*.

Figure 836. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

69.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

69.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

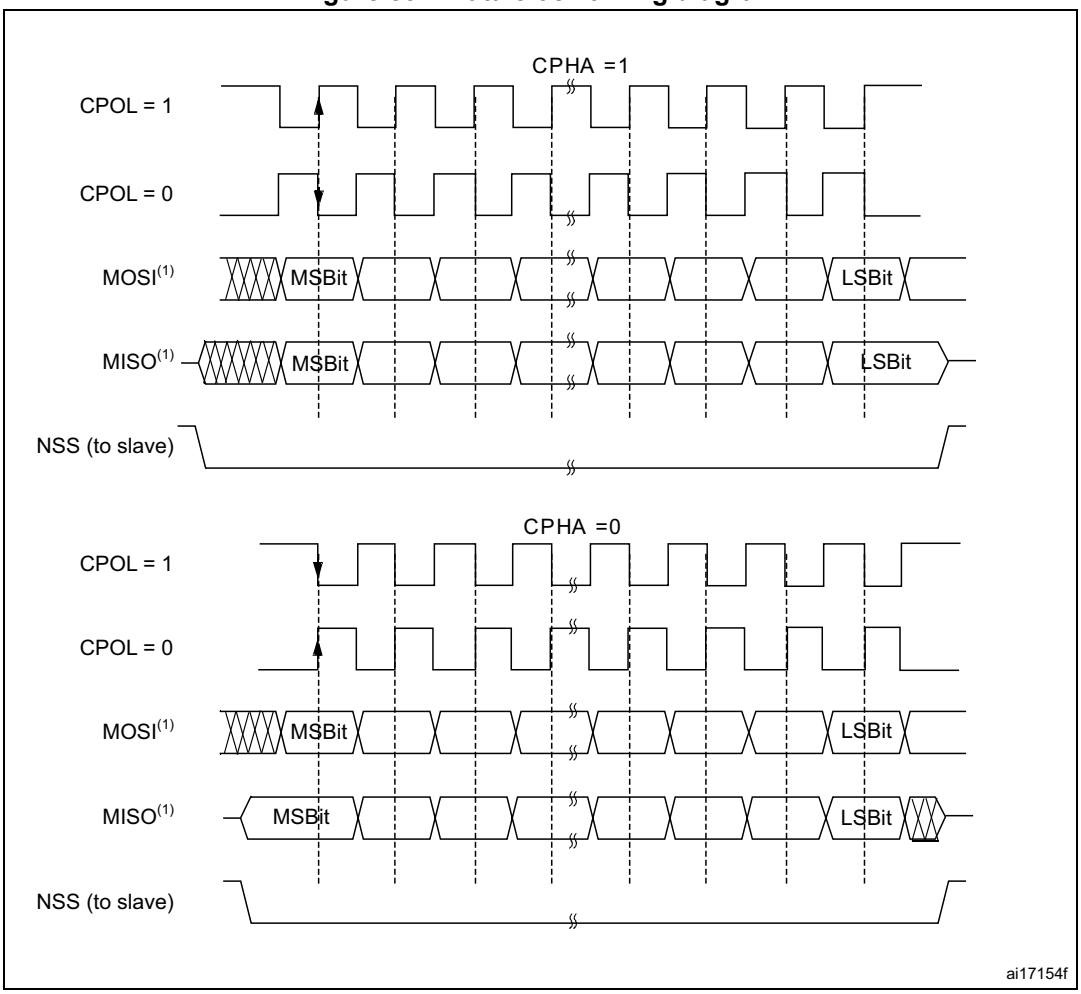
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 837*).

Figure 837 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 837. Data clock timing diagram



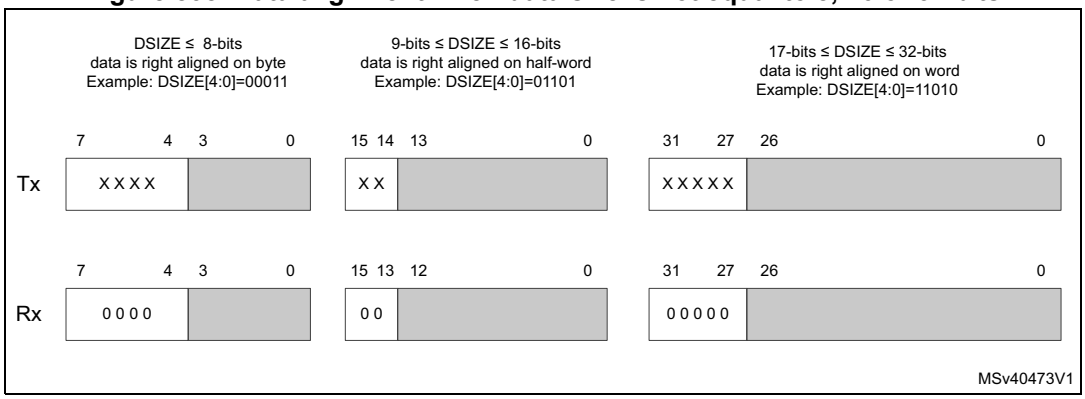
1. The order of data bits depends on the LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 69.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 838*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 69.4.12: SPI data transmission and reception procedures*.

Figure 838. Data alignment when data size is not equal to 8, 16 or 32 bits



69.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 69.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

69.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 69.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 69.4.14*.

69.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

69.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

69.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

69.4.15 Autonomous mode

The SPI is capable of handling and initializing transfers autonomously. It requires no specific system execution interaction until the ongoing transfer ends.

Such autonomous transfers can be handled not only in Run or Sleep mode but even in Stop mode when the SPI logic is able to provide temporary clock requests addressed to the reset

and clock controller (RCC) to ensure clocking of those SPI domains just necessary for handling the data flow between the memory and the peripheral interface at dependency in the SPI mode.

In Stop mode, the APB clock is requested by the peripheral each time the SPI registers need to be updated based on specific traffic events (mainly TXP and RXP). The required clock is provided by RCC if SPI autonomous mode is enabled at the RCC configuration and the SPI is clocked by an internal oscillator available in Stop mode.

Interrupts or DMA requests are then generated, depending on the SPI configuration. If no interrupt is enabled, the device remains in Stop mode. If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the device remains in Stop mode. If an enabled interrupt occurs, the device wakes up from Stop mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt is set. This is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low-power consumption especially, and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode.

Slave mode

When the SPI is configured as a standard slave and the device is at Stop mode, the SPI kernel clock and the SPI APB clock are not provided permanently. All the data flow between the SPI interface and associated FIFOs is handled by an external SCK clock provided by the outer master device within the serial interface clock domain. APB clock temporal requests are then based upon specific traffic events at dependency on the SPI configuration. As the slave never initializes a transfer, there is no need to synchronize any transfer start in this mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt stays set. This is why it is important to service these pending requests and clear their flag as soon as possible to achieve the lowest power consumption, and the application must acknowledge all pending interrupt events before switching the SPI to low-power mode.

Master mode

The SPI operating in master mode provides the SCK signal for outer slaves until the transfer is complete. The SCK signal is derived from the SPI clock generator running within the kernel clock domain fed from the RCC upon kernel clock request provided by the SPI when the device is in Stop mode. Temporal requests for the APB clock are then based upon specific traffic events, depending on the SPI configuration. The SPI master always initializes a transfer.

To minimize consumption in Stop mode, it is suggested to combine communication starts triggered by hardware (the TRIGEN bit is set in the SPI_AUTOCR register) and transfers of predefined data size (TSIZE > 0 in the SPI_CR2 register). This ensures that any APB clock request is suppressed between EOT handling and the next trigger event.

A transfer starts once the CSTART bit is set. In the case of master transmitter, the TxFIFO must be filled by data, too. The CSTART bit can be written either by software or by hardware when a synchronous trigger is detected at Run, Sleep, or Stop mode. The trigger source is selected by the TRIGSEL bits and enabled by the TRIGEN bit of the SPI_AUTOCR register. When the enabled trigger is detected, the transfer starts and continues by handling data until the EOT event or the transfer suspension. When the TRIGEN bit is changed, the user must prevent any trigger event occurrence. If the user cannot prevent that, the TRIGEN bit

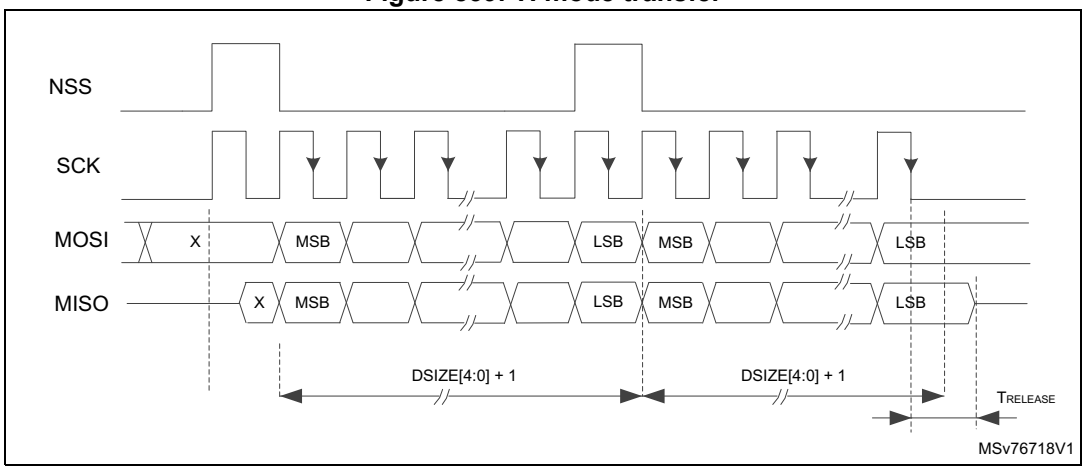
must be written while the SPI is disabled otherwise the peripheral behavior is not guaranteed.

69.5 SPI specific modes and control

69.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 839*.

Figure 839. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

69.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

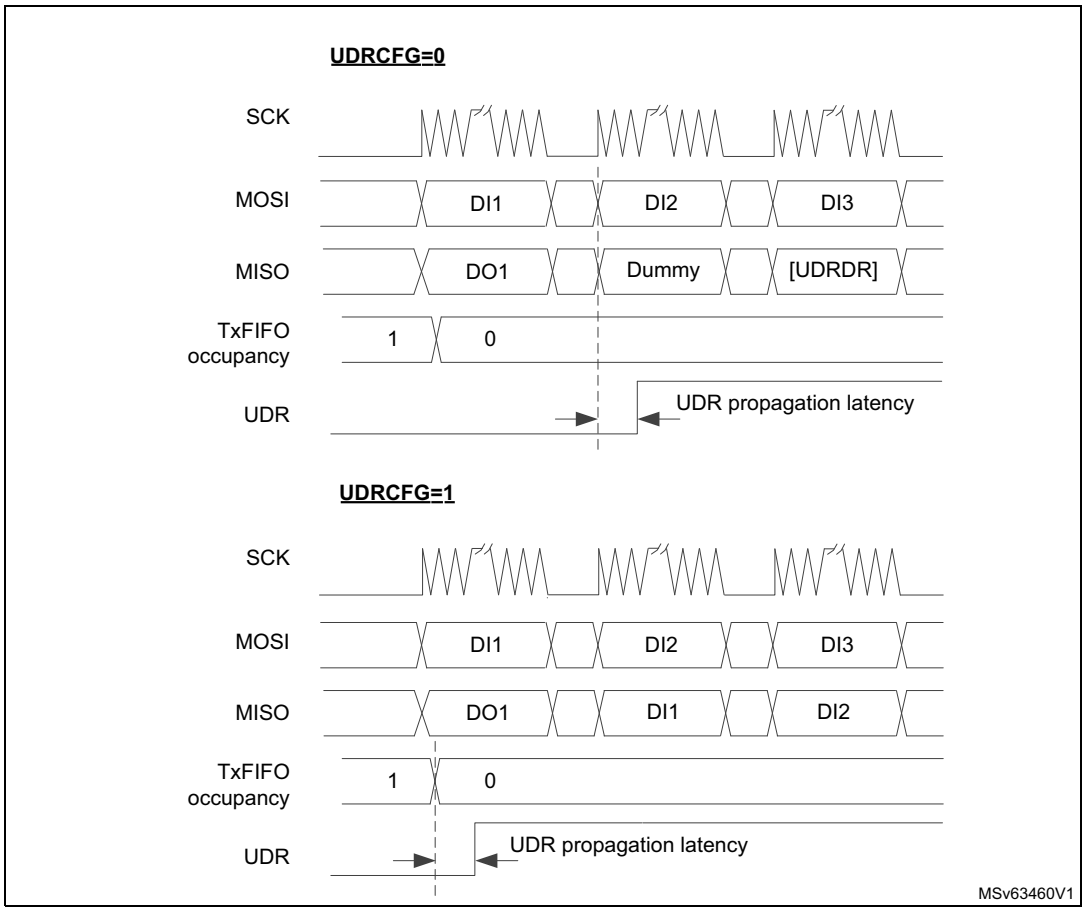
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 840*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 832*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 840. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

69.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 69.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

69.6 SPI in low-power modes

The SPI supports autonomous operation down to Stop mode, refer to *Section 69.4.15: Autonomous mode*.

Table 707. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the SPI registers is kept. If the autonomous mode is enabled at RCC configuration, and the SPI is clocked by an internal oscillator available in Stop mode, transfers are functional. The DMA requests are functional, and the interrupts in this mode cause the device to exit Stop mode.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 69.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

69.7 SPI interrupts

Table 708 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all

the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 708. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability(1)(2)
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC(3)		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 69.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

69.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

69.9 I2S functional description

69.9.1 I2S general description

The block diagram shown on *Figure 827* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 69.6: SPI in low-power modes* and *Section 69.10: I2S interrupts*).

a. Not always available, refer to *Section 69.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

69.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 3805*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 3805*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

69.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 709* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 709. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S autonomous mode control register (SPI_AUTOCR)	The complete register	-
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 709. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

69.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

69.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

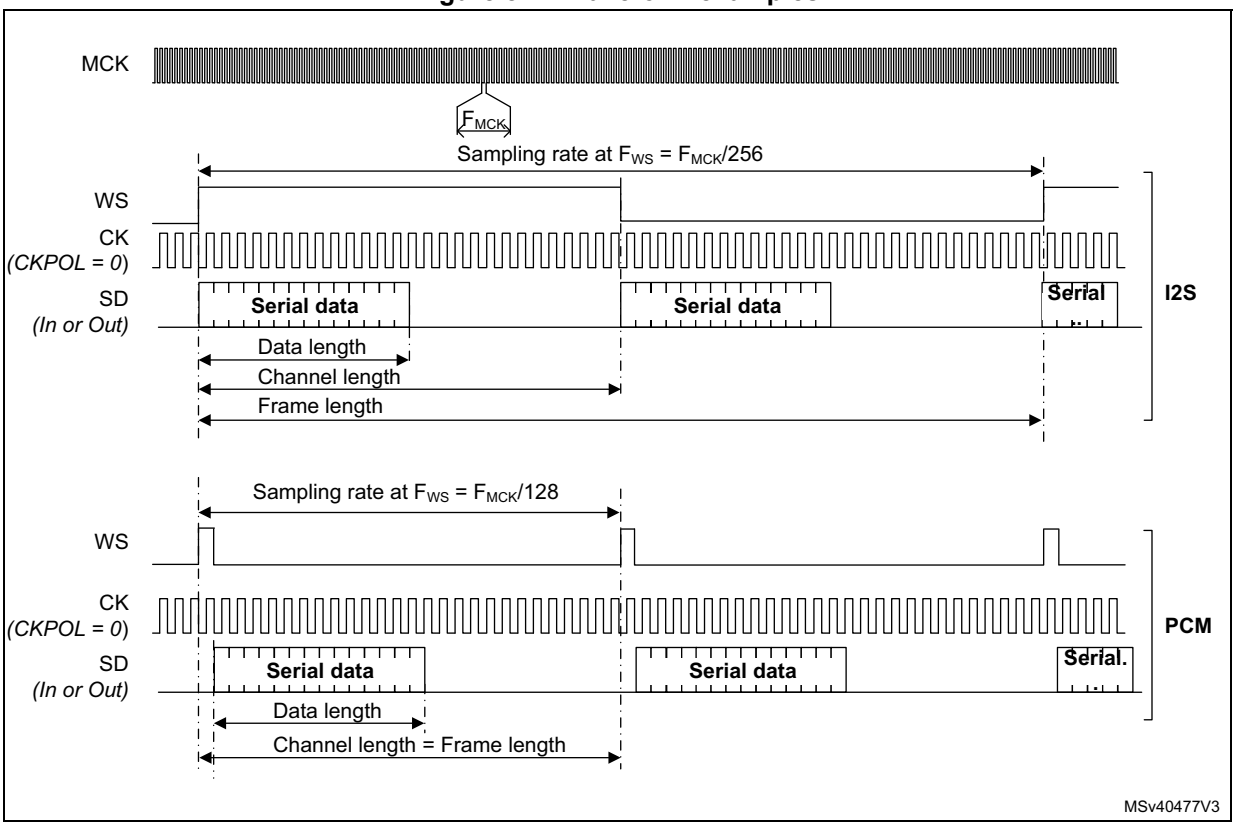
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]
- 32-bit data packed in a 32-bit channel[(a)]

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 841. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

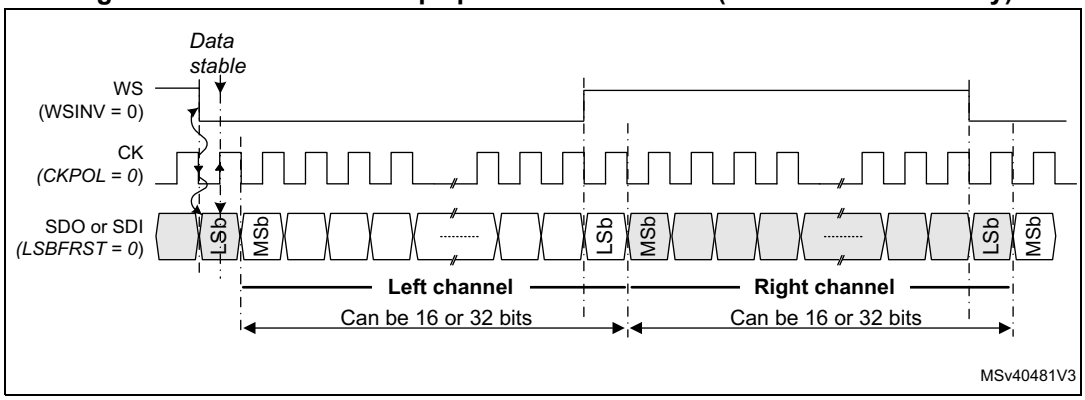
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 69.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 842. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



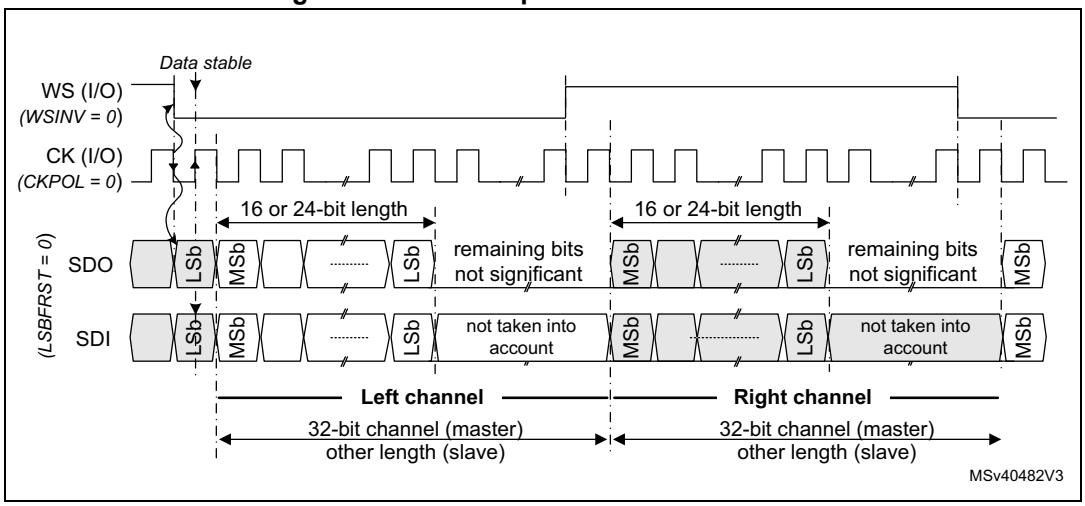
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 842 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 843. I2S Philips standard waveforms



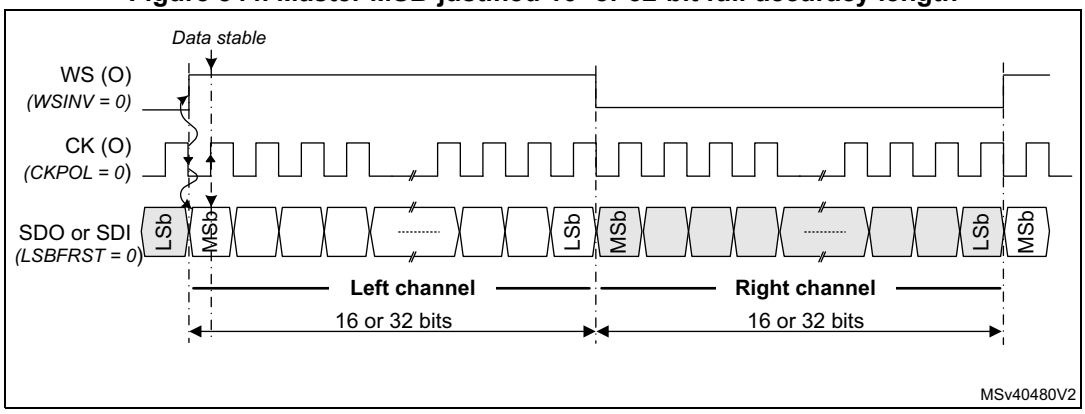
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 844. Master MSB-justified 16- or 32-bit full-accuracy length

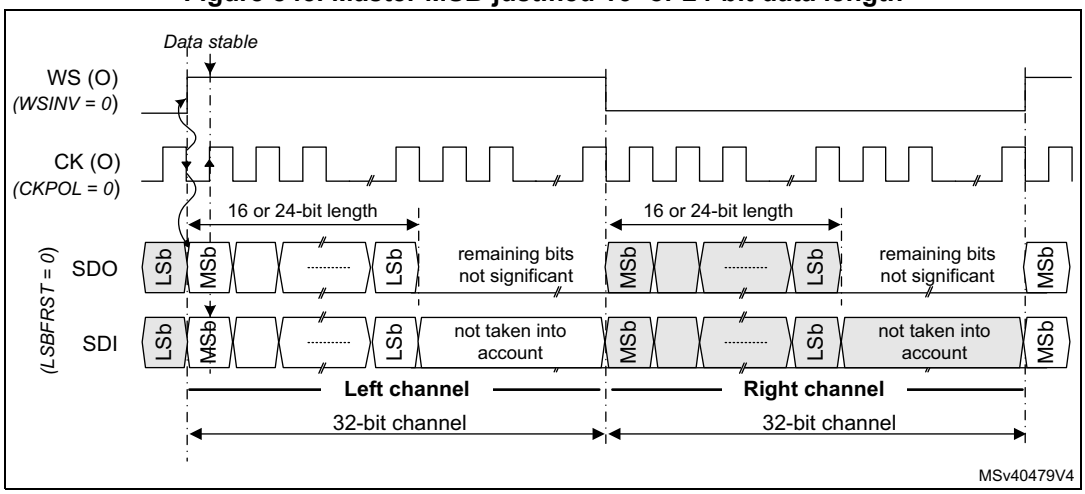


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

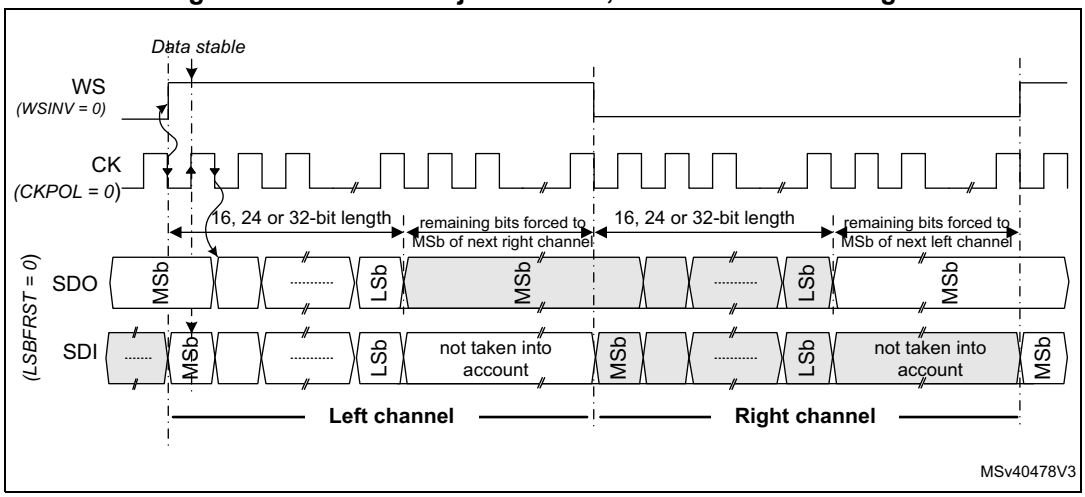
Figure 845. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 846*).

Figure 846. Slave MSB-justified 16-, 24- or 32-bit data length



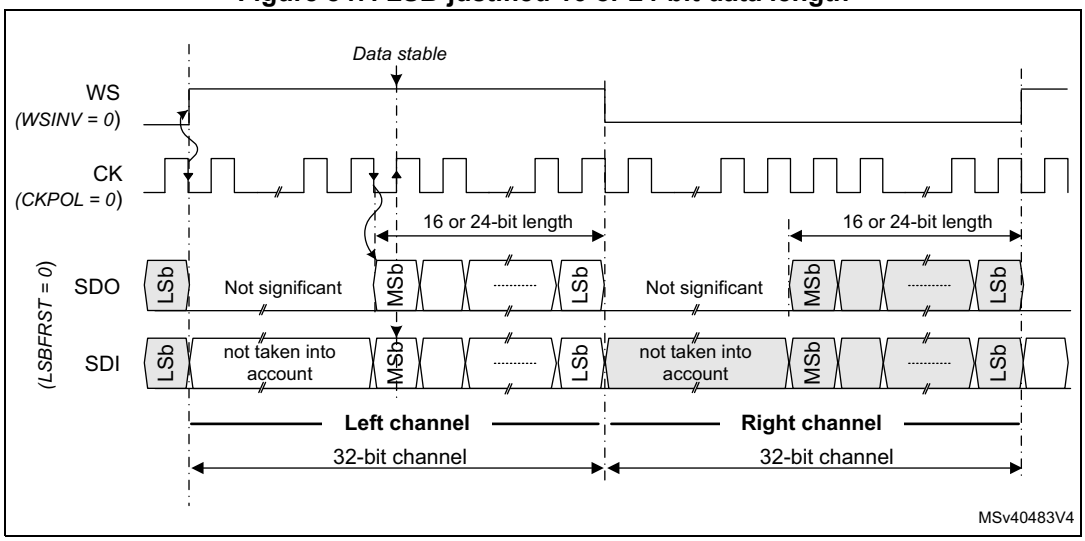
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 844*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 847. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

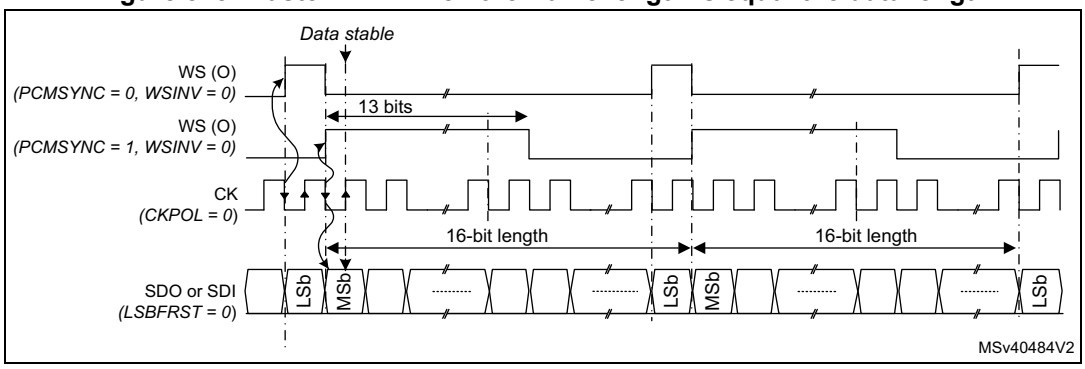
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

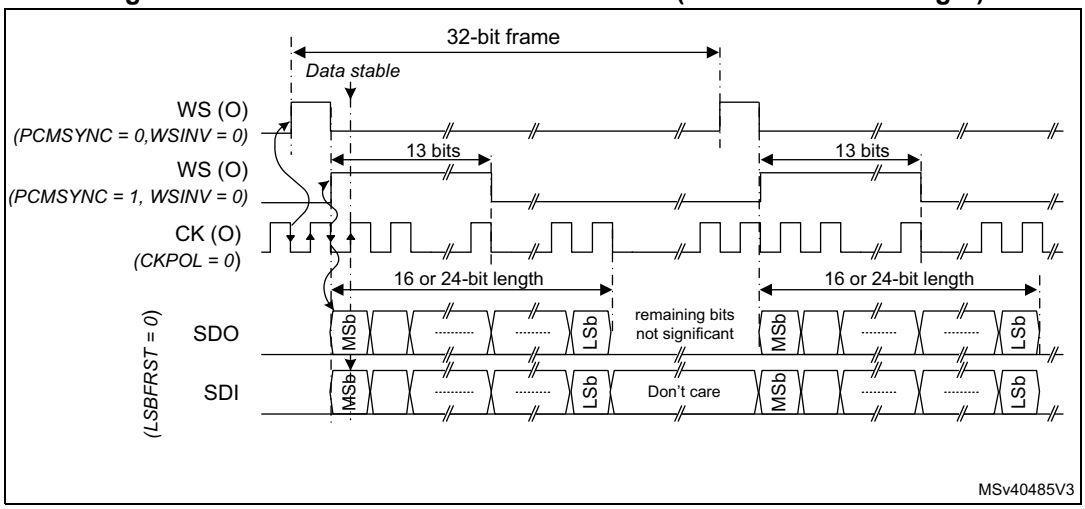
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 848. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

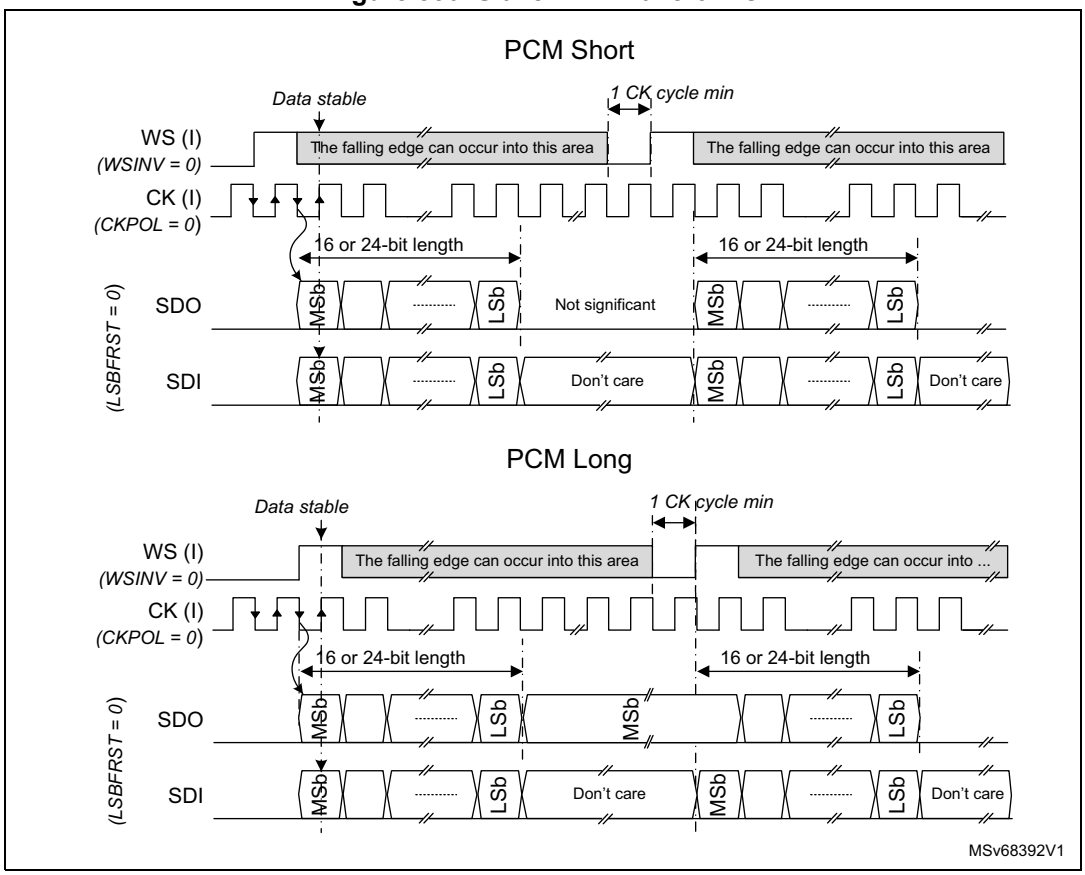
Figure 849. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 850*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 850. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

69.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 710 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 710. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 711* gives details on this feature.

Table 711. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

69.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

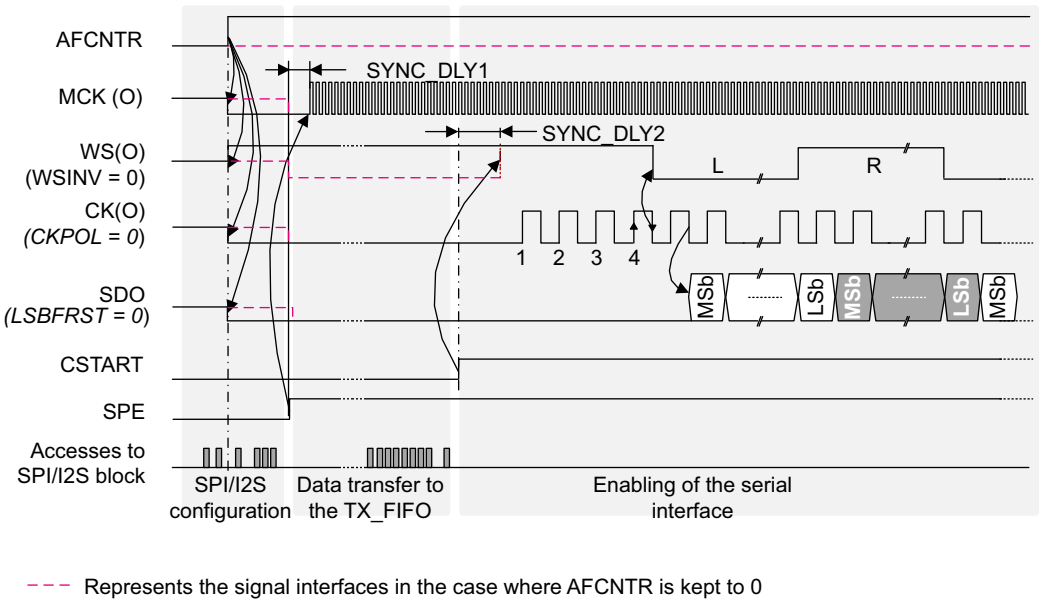
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 69.9.17* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 851*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1

and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 851. Startup sequence, I2S Philips standard, master



1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in *Section 69.9.9: Clock generator* for more information.
2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

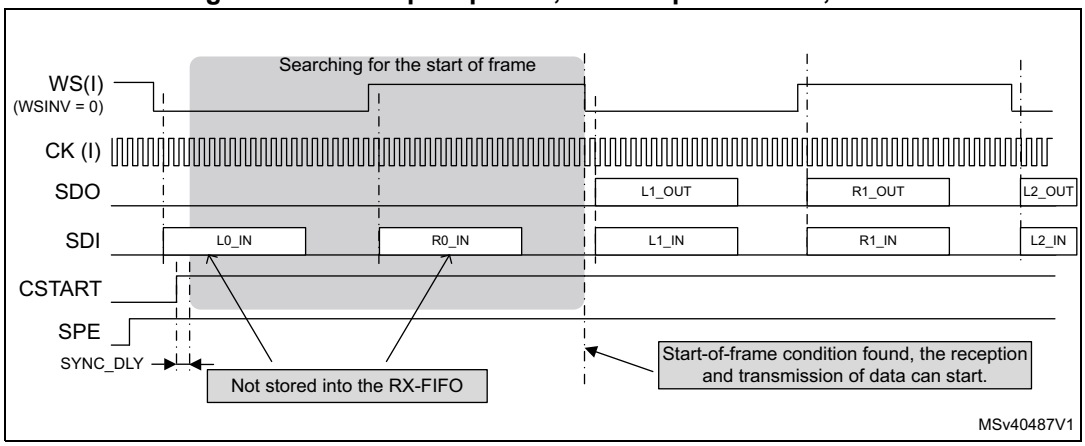
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 852 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 852. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

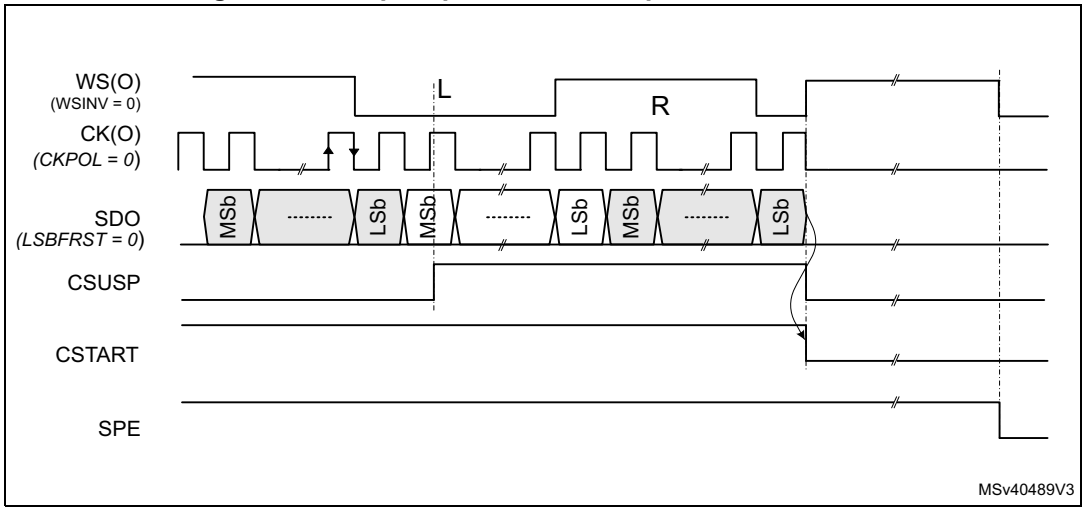
69.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 853 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 853. Stop sequence, I2S Philips standard, master

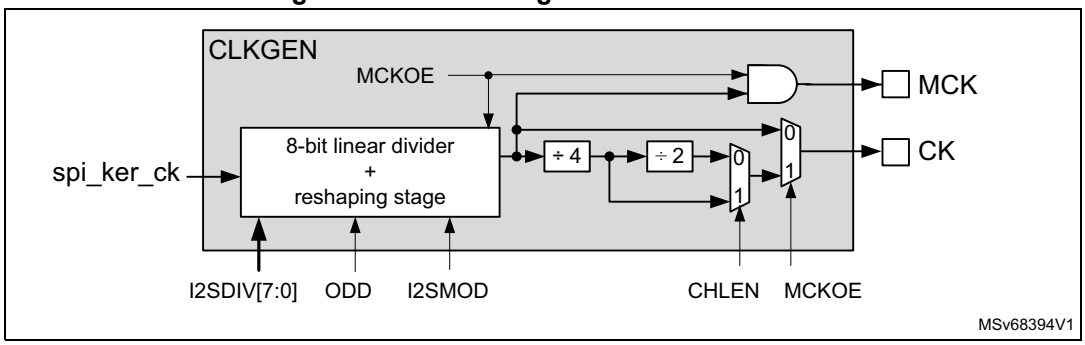


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

69.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 854. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of the MCK or the CK signal is not 50%. Care must be taken when an odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Table 712 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,
- SPE must be set

Table 712. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 712. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

69.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

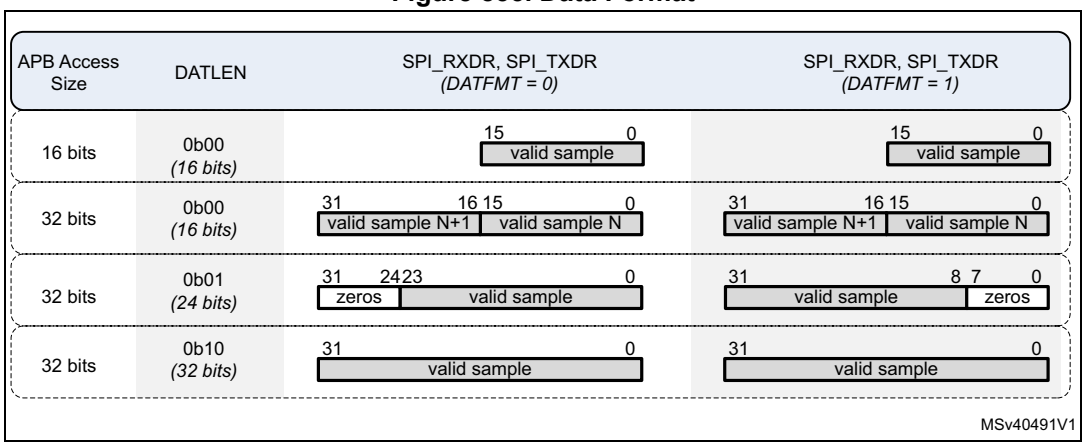
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 855 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 855. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 69.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 69.9.11* and *Section 69.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

69.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 69.10* for additional information on the interrupt function in I2S mode.

69.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

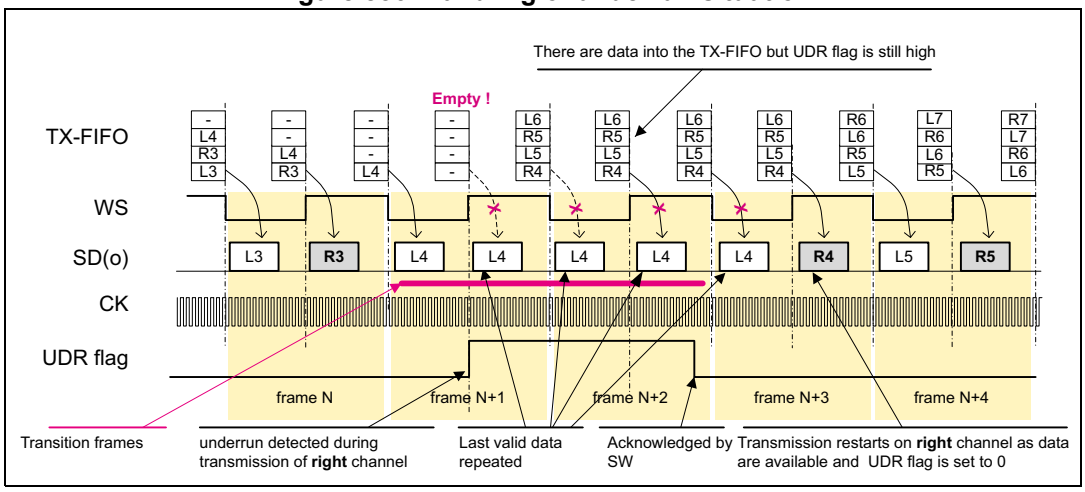
a. Not always available, refer to *Section 69.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 856. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

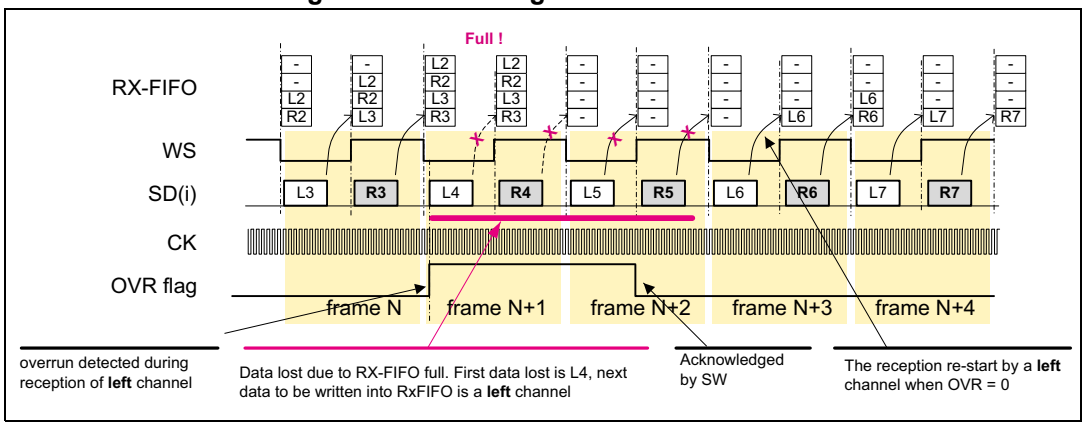
69.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO, and the OVR flag is cleared. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 857. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

69.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

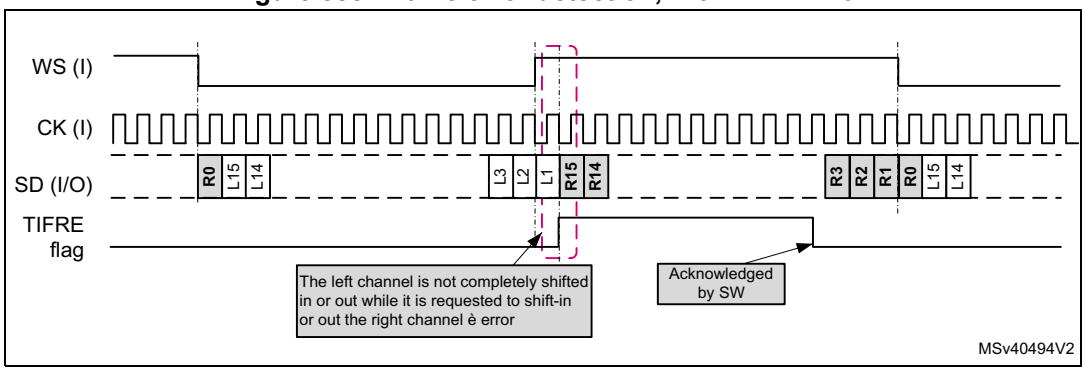
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 858 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 858. Frame error detection, with FIXCH = 0

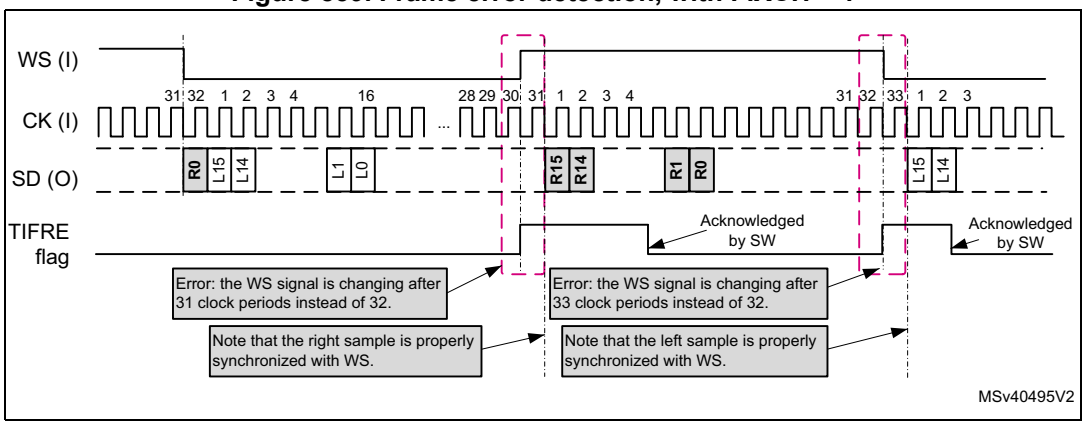


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 859*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 858* and *Figure 859* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 859. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

69.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

69.9.16 Autonomous mode

The I2S can request the kernel clock to the reset and clock controller (RCC), when a transfer is ongoing. So, even in Stop mode, it is possible to transfer samples from the TxFIFO to the serial interface, or from the serial interface to the RxFIFO. This mechanism only works when supported by the RCC and when an oscillator is selected as kernel clock.

When the APB clock is requested by the I2S, for example to transfer the samples from the memory to the TxFIFO or from the RxFIFO to the memory, a wake-up event is asserted. When the system is in Run, the APB clock is present, then the I2S asserts the interrupt line.

69.9.17 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. Clear all interrupt enable fields located in the SPI_IER register.
8. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
9. Set the SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An *Alternative sequence* is proposed hereafter.
10. When the application wants to start the data transfer:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Alternative sequence

- Steps 1 to 8 are similar to the previous sequence.
- If the data transfer uses DMA:
  - Program the DMA peripheral: two channels, one for RX and one for TX
  - Initialize the memory buffer with valid audio samples for the TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,

- If the data transfer is done via interrupt:
  - Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
- Set SPE, as soon as this bit is set to one the following actions may happen:
  - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  - If the DMA transfer is enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  - The MCK is generated. The application can then configure the external audio codec via an I2C interface.
- Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   - Program the DMA peripheral: one RX channel
   - Enable the DMA channel,
   - In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.

8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

69.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 713*.

Interrupt events can be enabled and disabled separately.

Table 713. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

69.11 SPI/I2S registers

69.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os
This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs
When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).
0: AF configuration is not locked
1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter
0: all zero pattern is applied
1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver
0: All zero pattern is applied
1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration
0: Full size (33-bit or 17-bit) CRC polynomial is not used
1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode
In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.
0: SPI is receiver
1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request
This bit reads as zero.
In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.
The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.
After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start
This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 69.9.8: Stop sequence*.
In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.
0: master transfer is at idle
1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, part of the SPI_AUTOCR register and IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

69.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

69.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator

0: bypass is disabled

1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting

000: SPI master clock/2

001: SPI master clock/4

010: SPI master clock/8

011: SPI master clock/16

100: SPI master clock/32

101: SPI master clock/64

110: SPI master clock/128

111: SPI master clock/256

Note: MBR setting is considered at slave working at TI mode, too (see Section 69.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable

0: CRC calculation disabled

1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.

The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable

0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable

0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 840: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

69.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

69.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable

0: MODF interrupt disabled

1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable

0: TIFRE interrupt disabled

1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable

0: CRC interrupt disabled

1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable

0: OVR interrupt disabled

1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

69.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus or during autonomous operation in low-power mode.
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

69.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

69.11.8 SPI/I2S autonomous mode control register (SPI_AUTOCR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Hardware control of CSTART triggering enable

0: Hardware control disabled

1: Hardware control enabled

Note: if user cannot prevent trigger event during write, the TRIGEN must be changed when SPI is disabled

Bit 20 **TRIGPOL:** Trigger polarity

0: trigger is active on raising edge

1: trigger is active on falling edge

Note: This bit can be written only when SPE = 0.

Bits 19:16 **TRIGSEL[3:0]**: Trigger selection (refer *Description of SPI interconnections*).

0000: spi_trg0 is selected

0001: spi_trg1 is selected

...

1111: spi_trg15 is selected

Note: these bits can be written only when SPE = 0.

Bits 15:0 Reserved, must be kept at reset value.

69.11.9 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

69.11.10 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

69.11.11 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

69.11.12 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

69.11.13 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

69.11.14 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

69.11.15 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 69.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 69.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 69.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 69.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

69.11.16 SPI/I2S hardware configuration register (SPI_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPBCFG[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **OPBCFG[7:0]:** Number of option bits available
0000 0000: No option bit is available.
0000 0001: One option bit is available.
....
0001 1111: 31 option bits are available.
0010 0000: 32 option bits are available.
Others: Reserved.

69.11.17 SPI/I2S hardware configuration register (SPI_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0111 1144 (SPI1_2_3)
Reset value: 0x0110 0133 (SPI4_5_6)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FULLCFG[3:0]				TRGCFG[3:0]				DSCFG[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2SCFG[3:0]				CRCCFG[3:0]				RXFCFG[3:0]				TXFCFG[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **FULLCFG[3:0]:** SPI full-featured version configuration
0000: Limited-featured version is applied.
0001: Full-featured version is applied.
Others: Reserved.

Bits 23:20 **TRGCFG[3:0]:** Autonomous trigger feature configuration
0000: Trigger feature is disabled at autonomous mode.
0001: Trigger feature is enabled at autonomous mode.
Others: Reserved.

Bits 19:16 **DSCFG[3:0]:** SPI data size configuration
All the other combinations are reserved.
0000: maximum data size is 16 bits
0001: maximum data size is 32 bits

Bits 15:12 **I2SCFG[3:0]:** I2S configuration
All the other combinations are reserved.
0000: I2S support is not implemented
0001: I2S support is implemented

Bits 11:8 **CRCCFG[3:0]:** CRC configuration for SPI
All the other combinations are reserved.
0000: CRC support is not implemented
0001: CRC support is implemented

Bits 7:4 **RXFCFG[3:0]:** RxFIFO size
All the other combinations are reserved.
0002: the FIFO size is 4 bytes
0003: the FIFO size is 8 bytes
0004: the FIFO size is 16 bytes
0005: the FIFO size is 32 bytes

Bits 3:0 **TXFCFG[3:0]:** TxFIFO size
All the other combinations are reserved.
0002: the FIFO size is 4 bytes
0003: the FIFO size is 8 bytes
0004: the FIFO size is 16 bytes
0005: the FIFO size is 32 bytes

69.11.18 SPI/I2S version register (SPI_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** major version

Bits 3:0 **MINREV[3:0]:** minor version

69.11.19 SPI/I2S identification register (SPI_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification

69.11.20 SPI/I2S size identification register (SPI_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification
Bits[31:8]: fixed code 0xA3C5 DD0
Bits[7:0]: range of address decoding boundary
0x01: 1 Kbyte.
0x02: 2 Kbytes.
0x04: 4 Kbytes.
0x08: 8 Kbytes.
Others: Reserved.

69.11.21 SPI/I2S register map

Table 714. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	SPI_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0																
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 714. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]		I2SMOD	
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0
0x0E4 - 0x3E8	Reserved	Res.																															
0x3EC	SPI_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPBCFG[7:0]							
	Reset value	0x0000 0000																															
0x3F0	SPI_HWCFGR1	Res.	Res.	Res.	Res.	FULLCFG[3:0]				TRGCFG[3:0]				DSCFG[3:0]				I2SCFG[3:0]				CRCCFG[3:0]				RXFCFG[3:0]				TXFCFG[3:0]	
	Reset value	0x0111 1144 (SPI1_2_3) 0x0110 0133 (SPI4_5_6)																															
0x3F4	SPI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0022																															
0x3F8	SPI_IPIDR	ID[31:0]																															
	Reset value	0x0013 0023																															
0x3FC	SPI_SIDR	SID[31:0]																															
	Reset value	0xA3C5 DD01																															

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 69.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

70 Serial audio interface (SAI)

70.1 SAI introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

70.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Possible synchronization between multiple SAIs
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception

- Interrupt sources when enabled:
  - Errors
  - FIFO requests
- 2-channel DMA interface

70.3 SAI implementation

Table 715. SAI features(1)

SAI features	SAI1	SAI2	SAI3	SAI4
I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC'97	X	X	X	X
FIFO size	8 words	8 words	8 words	8 words
SPDIF	X	X	X	X
PDM	X(2)	-	-	X(2)

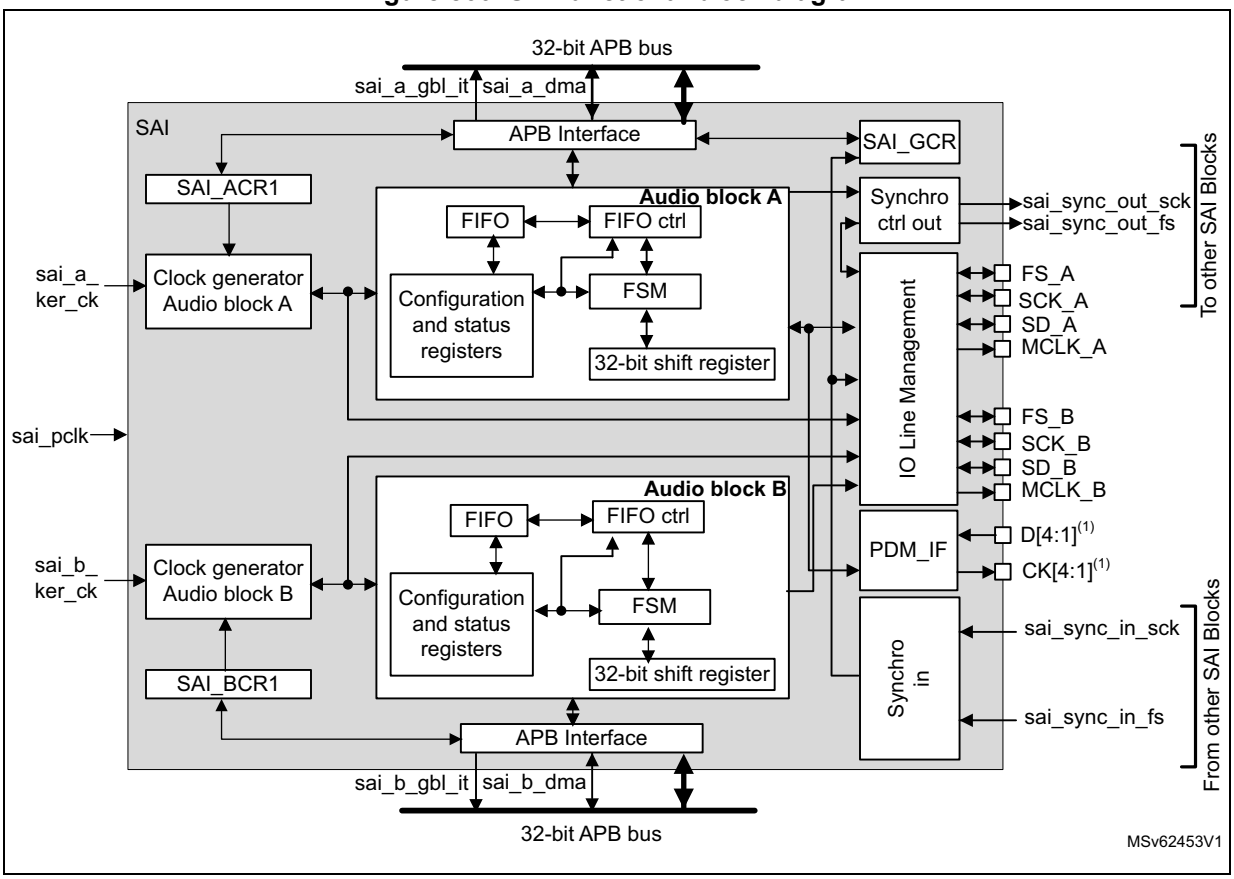
1. 'X' = supported, '-' = not supported.
2. Only signals D[4:1], and CK[2:1] are available.

70.4 SAI functional description

70.4.1 SAI block diagram

Figure 860 shows the SAI block diagram while *Table 716* and *Table 717* list SAI internal and external signals.

Figure 860. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 70.3: SAI implementation* for details.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

70.4.2 SAI pins and internal signals

Table 716. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 717. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A	Input/output	Audio block A bit clock
SAI_MCLK_A	Output	Audio block A master clock
SAI_FS_A	Input/output	Frame synchronization line for audio block A
SAI_SD_A	Input/output	Data line for block A
SAI_SCK_B	Input/output	Audio block B bit clock
SAI_MCLK_B	Output	Audio block B master clock
SAI_FS_B	Input/output	Frame synchronization line for audio block B
SAI_SD_B	Input/output	Data line for block B
SAI_CK[2:1]	Output	PDM bitstream clock(1)
SAI_D[4:1]	Input	PDM bitstream data(1)

1. These signals might not be available in all SAI instances. Refer to *Section 70.3: SAI implementation* for details.

70.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC’97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

70.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

External synchronization

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follows:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblocks is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: The SYNCIN[1:0] and SYNCOUT[1:0] bits are located in the SAI_GCR register, and the SYNCEN bits in the SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 3. Positions noted as ‘Reserved’ must not be used.

Table 718. External synchronization selection

Block instance	SYNCIN= 3	SYNCIN= 2	SYNCIN= 1	SYNCIN= 0
SAI1	SAI4	SAI3	SAI2	Reserved
SAI2	SAI4	SAI3	Reserved	SAI1
SAI3	SAI4	Reserved	SAI2	SAI1
SAI4	Reserved	SAI3	SAI2	SAI1

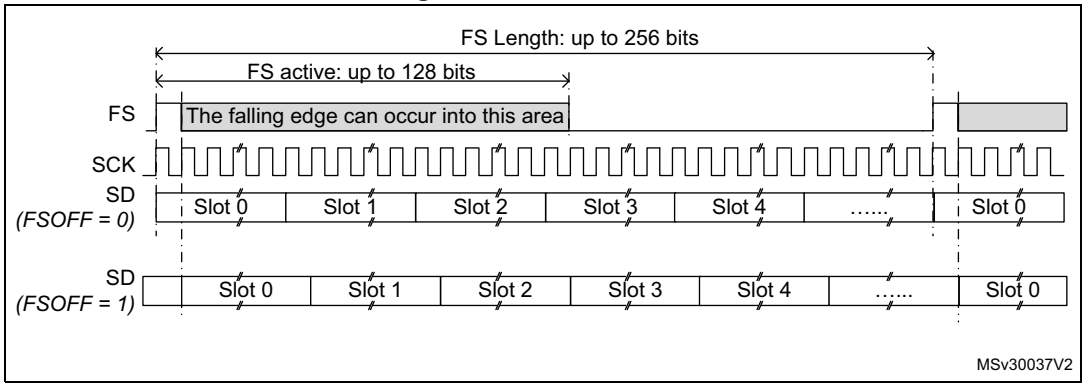
70.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

70.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 861* illustrates this flexibility.

Figure 861. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z

depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 70.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 70.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 70.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 70.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

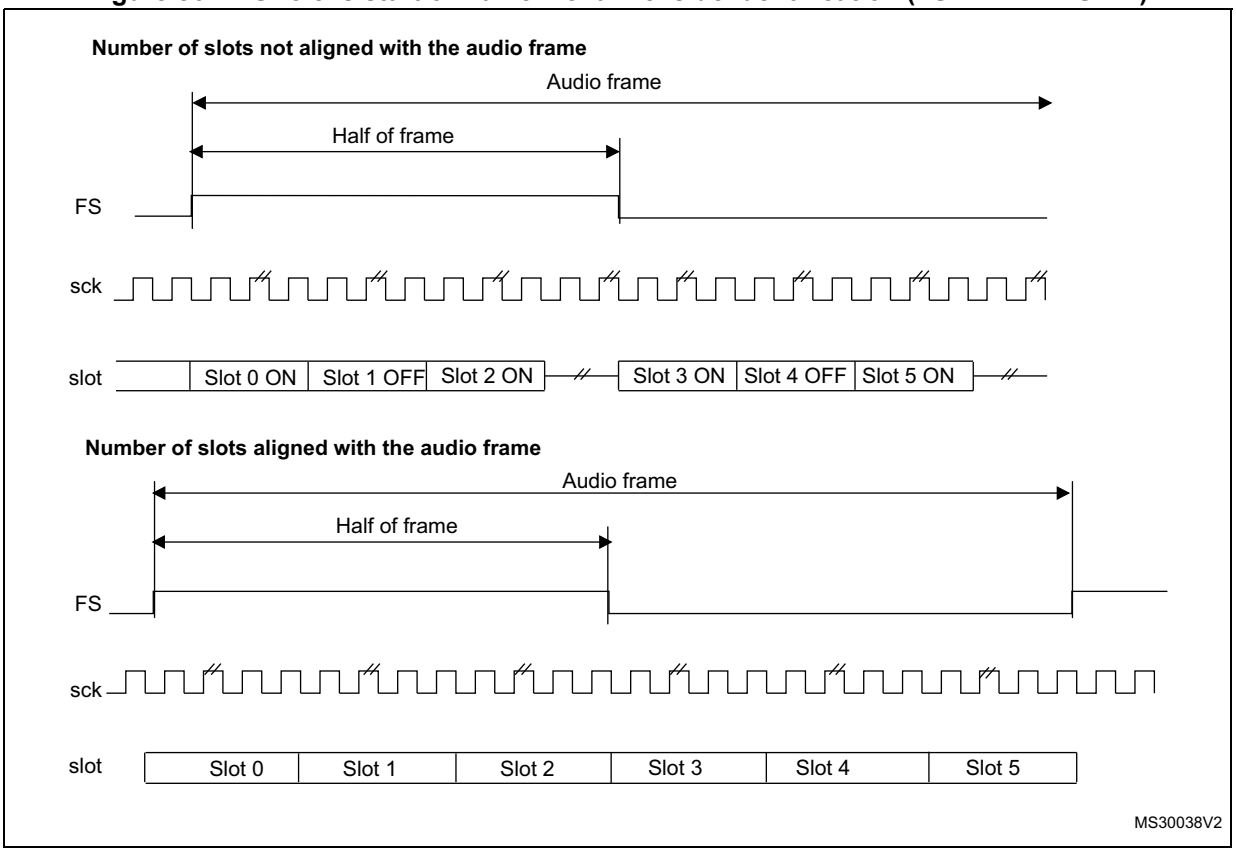
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 862. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

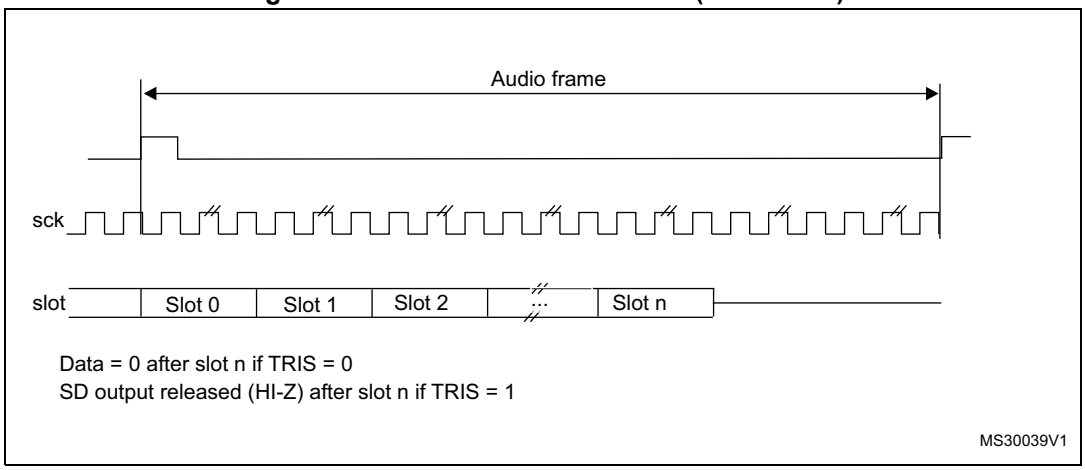


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 863. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

70.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

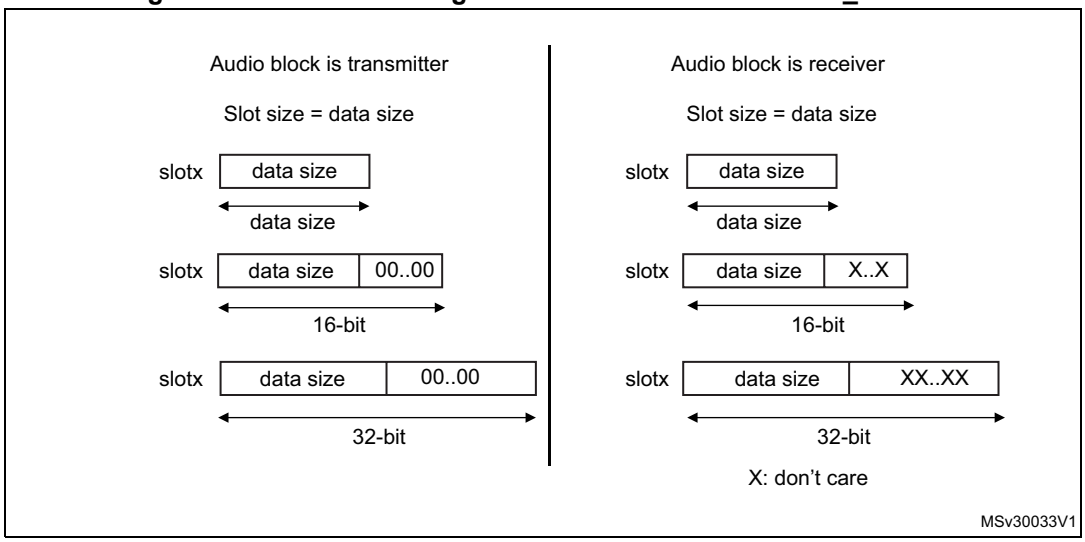
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

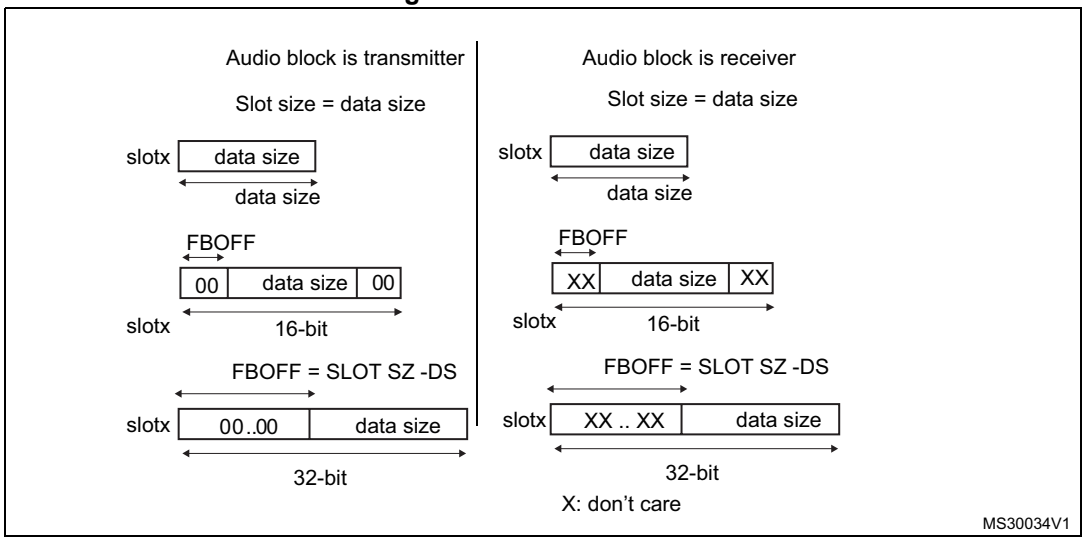
The slot size is also configurable as shown in *Figure 864.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 864. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 865. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS)
- DS ≤ SLOTSZ
- NBSLOT x SLOTSZ ≤ FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 70.4.11: AC'97 link controller*.

70.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 719 shows MCLK_x activation conditions.

Table 719. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

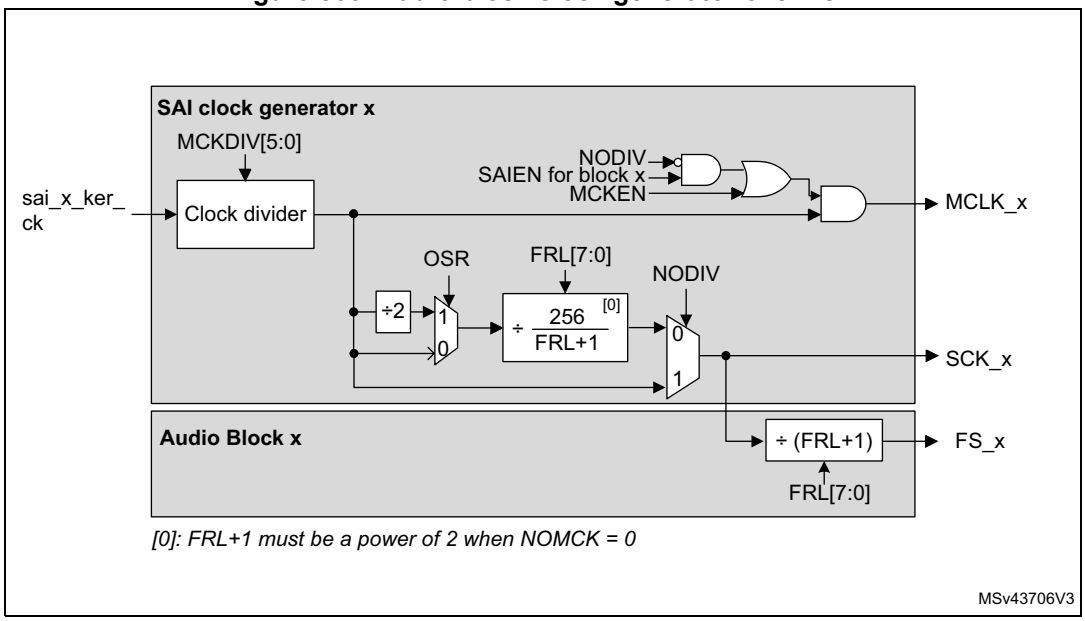
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 866 illustrates the architecture of the audio block clock generator.

Figure 866. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
- $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 720 gives programming examples for 48, 96, and 192 kHz.

Table 720. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

70.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

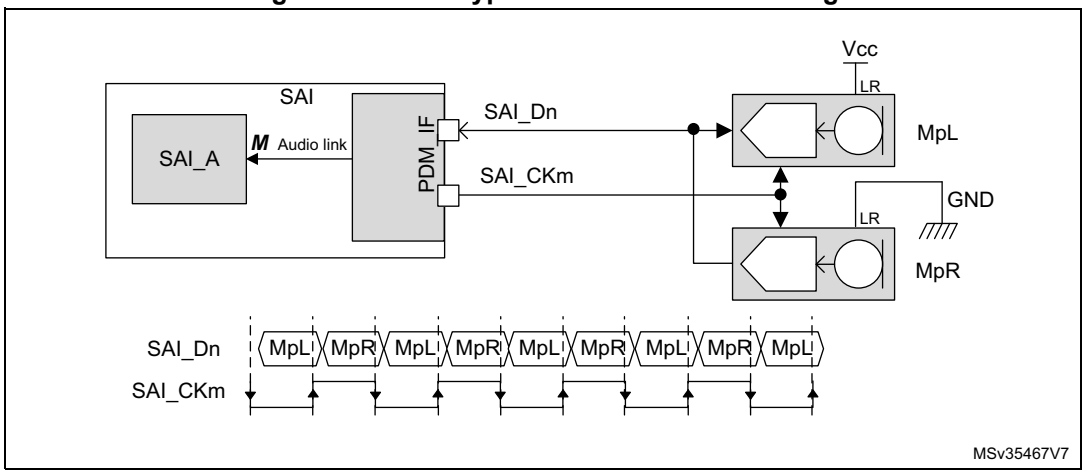
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

70.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 70.3: SAI implementation*).

Figure 867 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 867. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

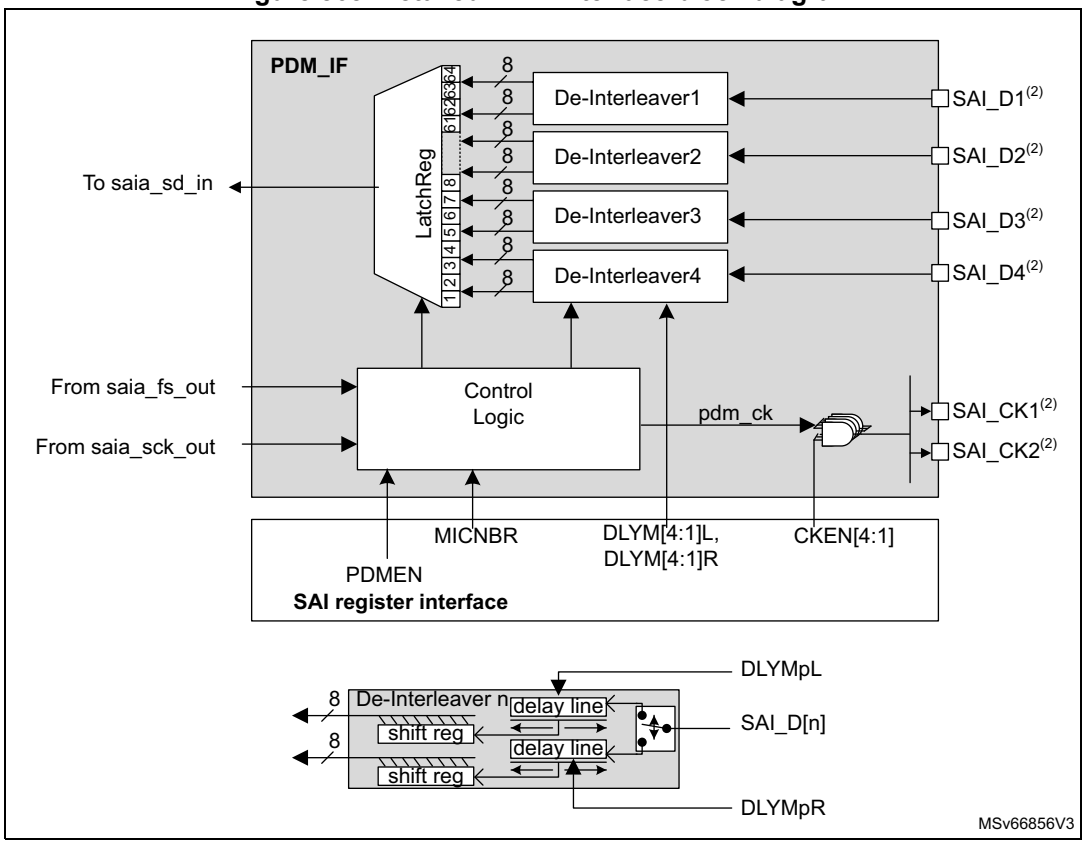
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 868 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 868. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 70.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose

between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

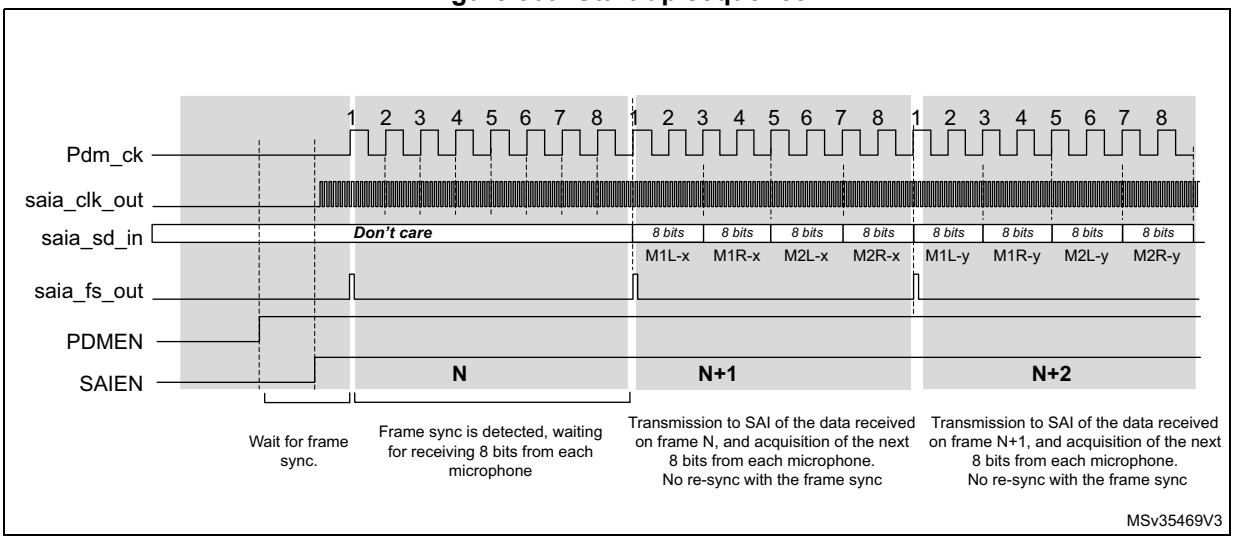
1. Configure SAI_A in TDM master mode (see *Table 721*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 869 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 869. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

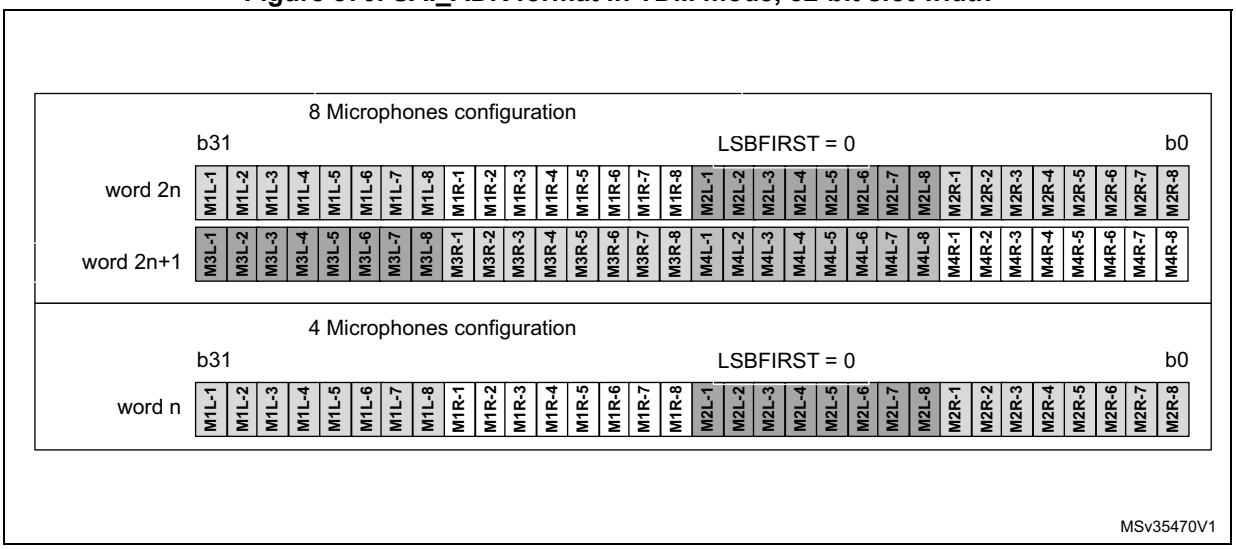
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 870*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 870. SAI_ADR format in TDM mode, 32-bit slot width

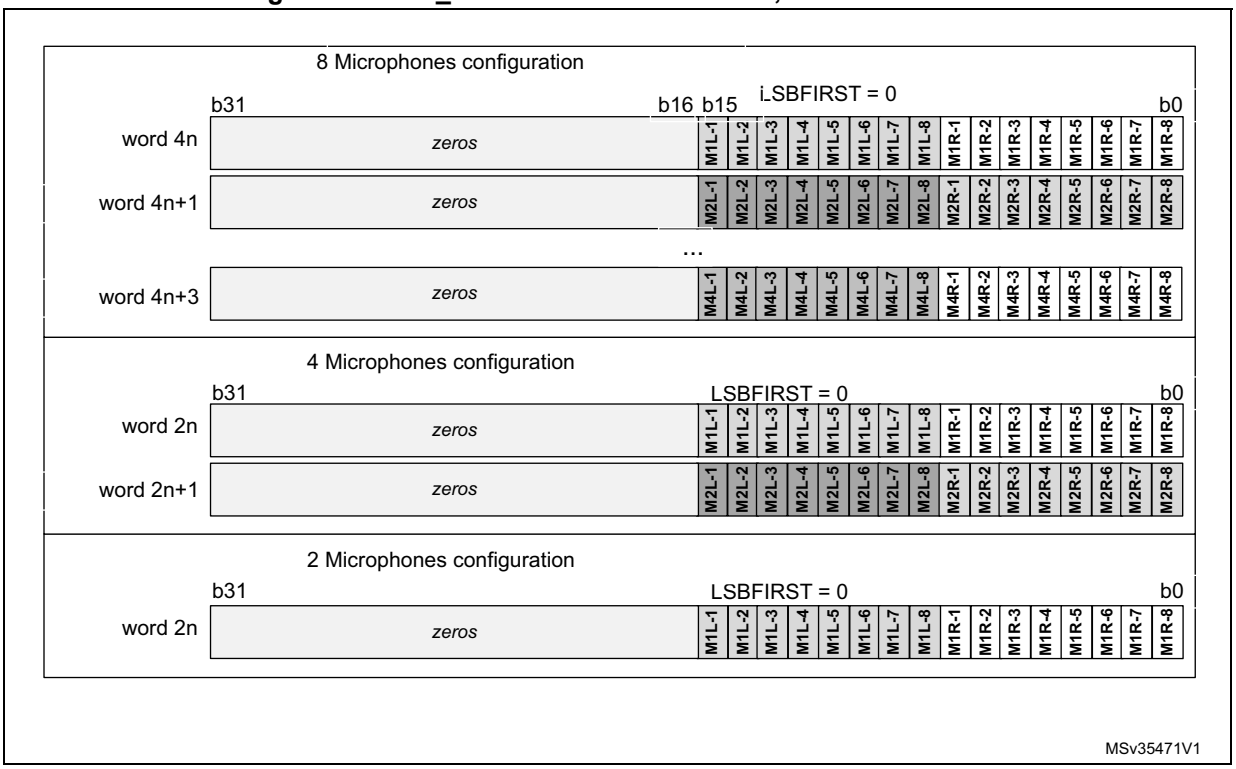


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 871*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 871. SAI_ADR format in TDM mode, 16-bit slot width

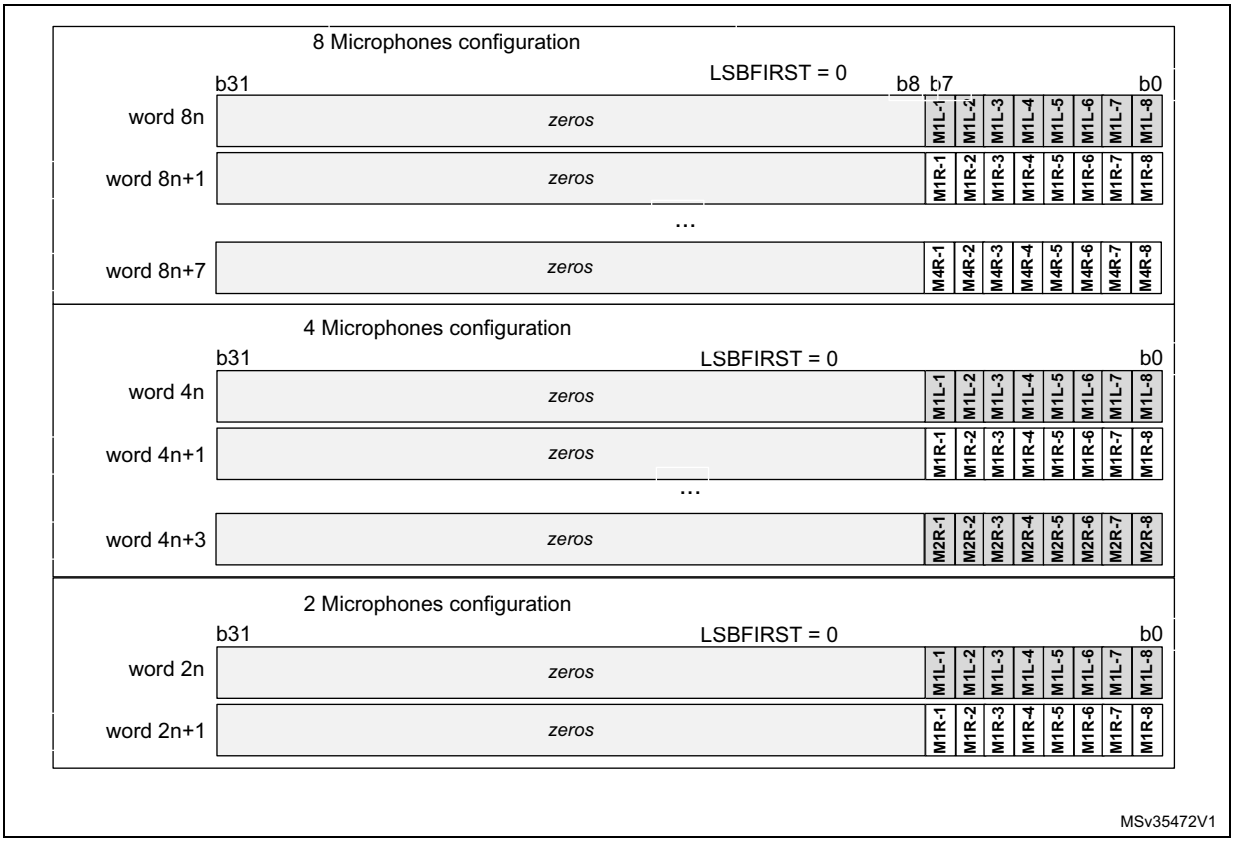


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 872*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 872. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 721* to ensure a good connection with the PDM interface.

Table 721. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 722*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 722*.
FSALL	0	Pulse width is one bit clock cycle.

Table 721. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 722*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 722*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 721*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 70.6.18*)

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 722. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 721: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device*.
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

70.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

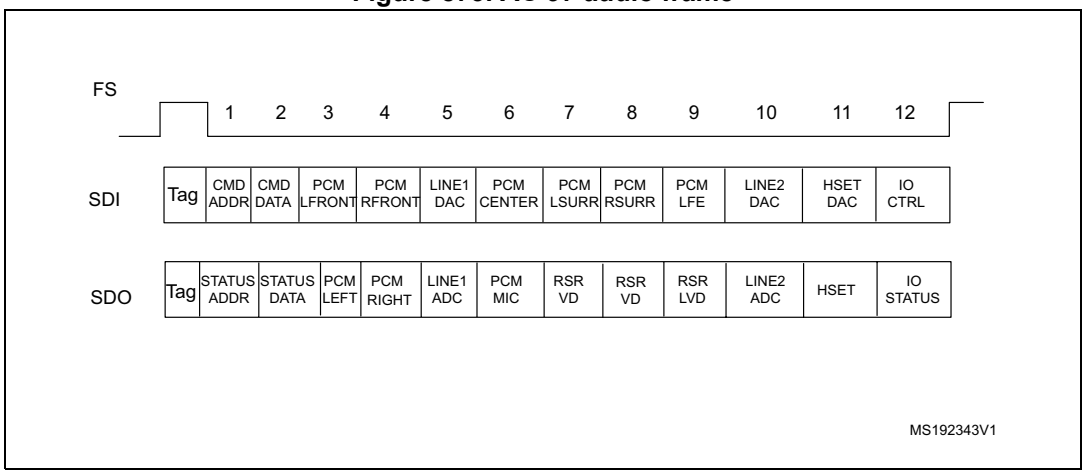
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 873 shows an AC’97 audio frame structure.

Figure 873. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

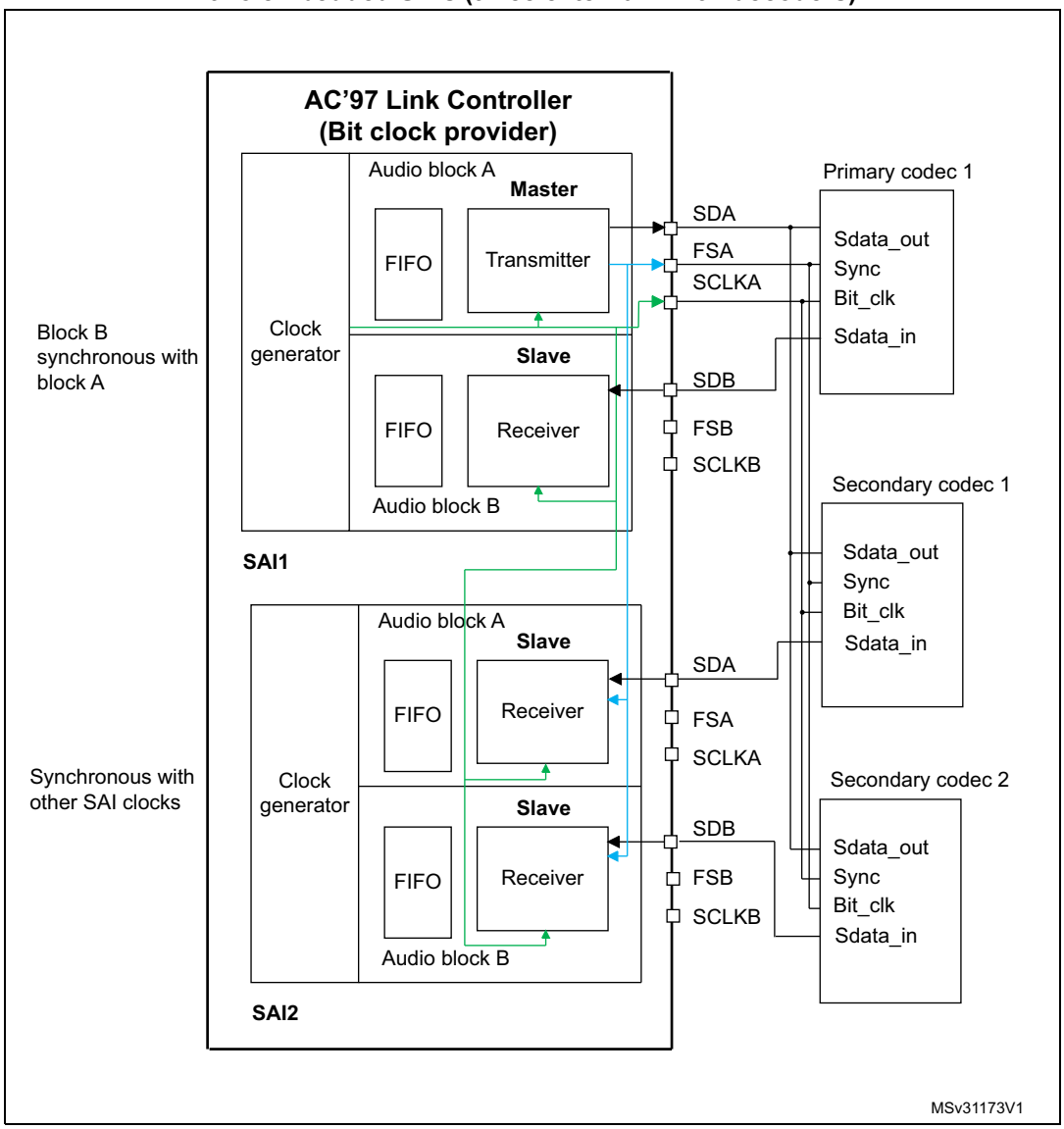
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 874*.

In SAI1, the audio block A must be declared as asynchronous master transmitter, whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 874. Example of typical AC'97 configuration on devices featuring at least two embedded SAIs (three external AC'97 decoders)



In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 70.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

70.4.12 SPDIF output

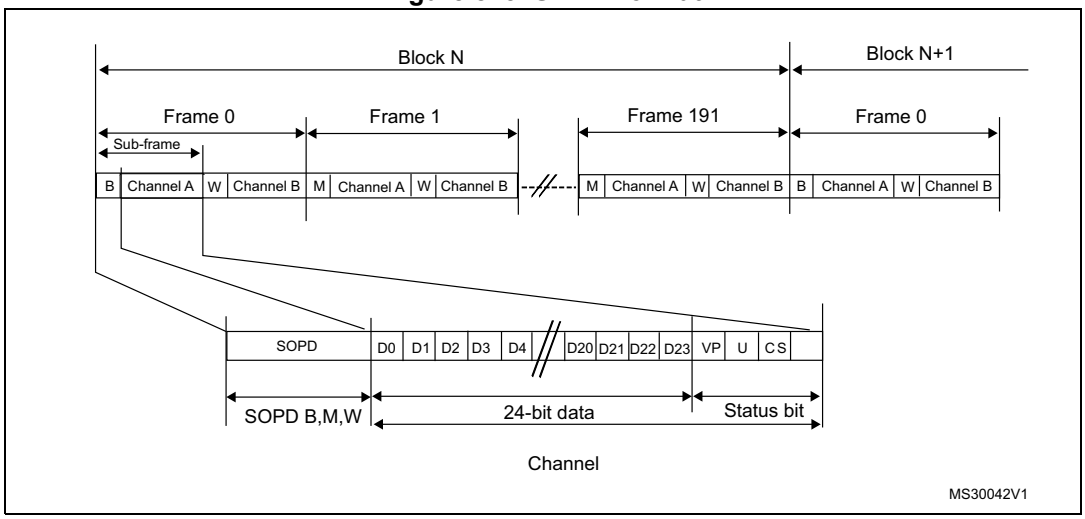
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 875*.

Figure 875. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 723*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 723. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

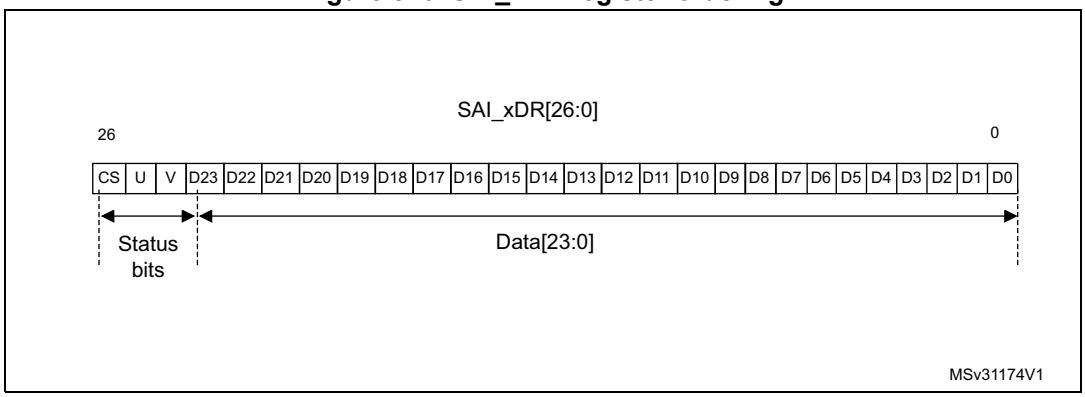
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 876. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 724*.

Table 724. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 725. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

70.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 877*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

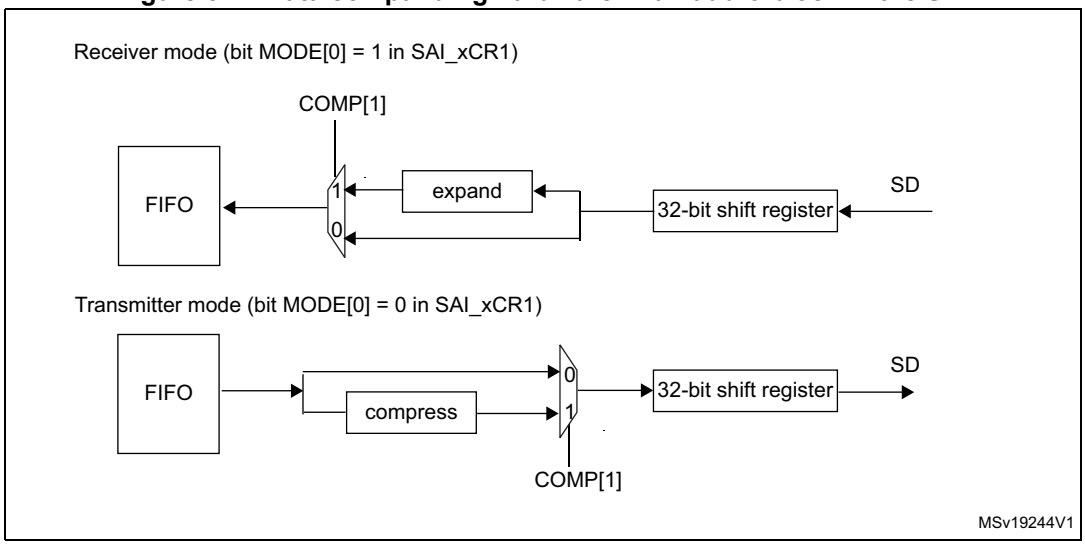
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1's complement or 2's complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 877. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

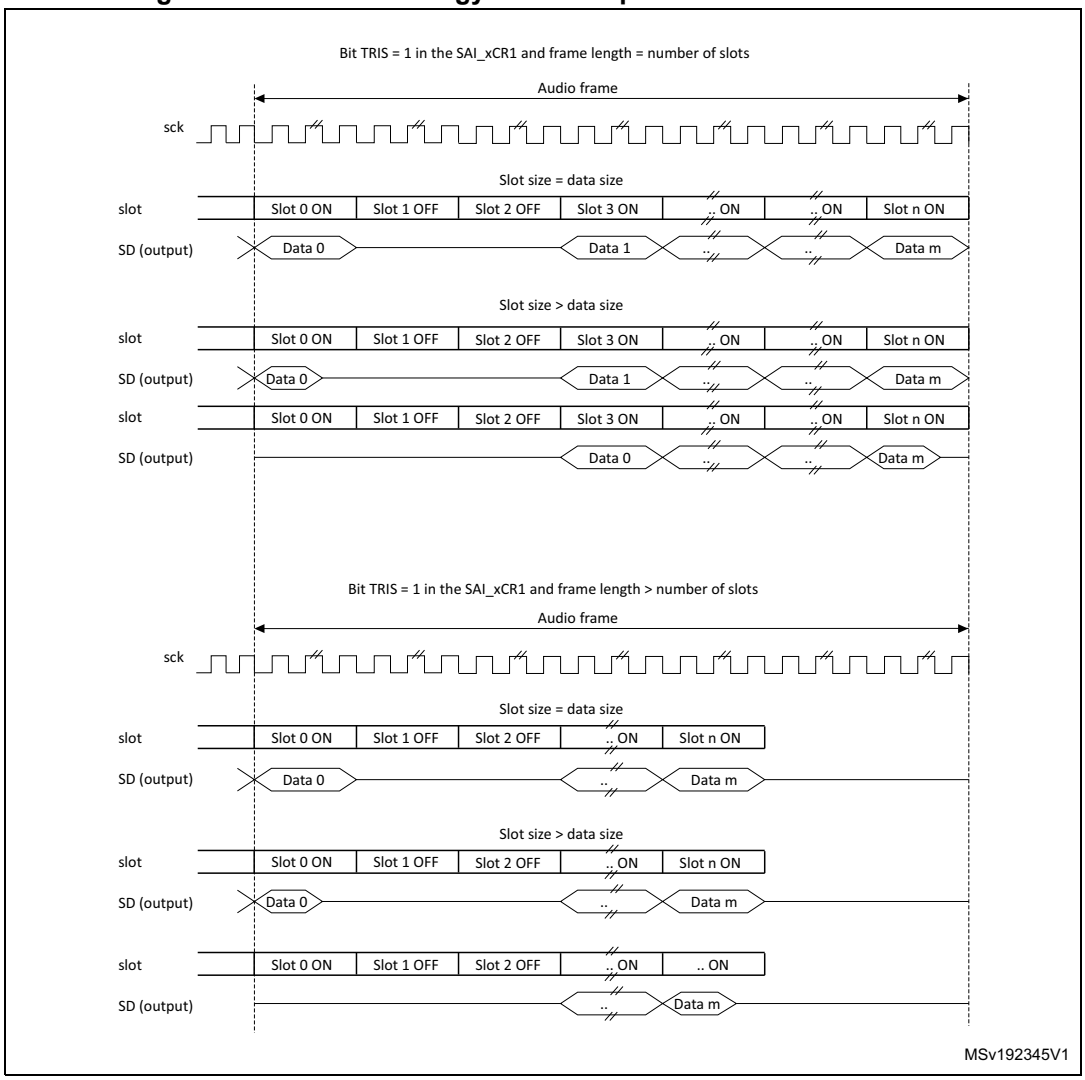
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

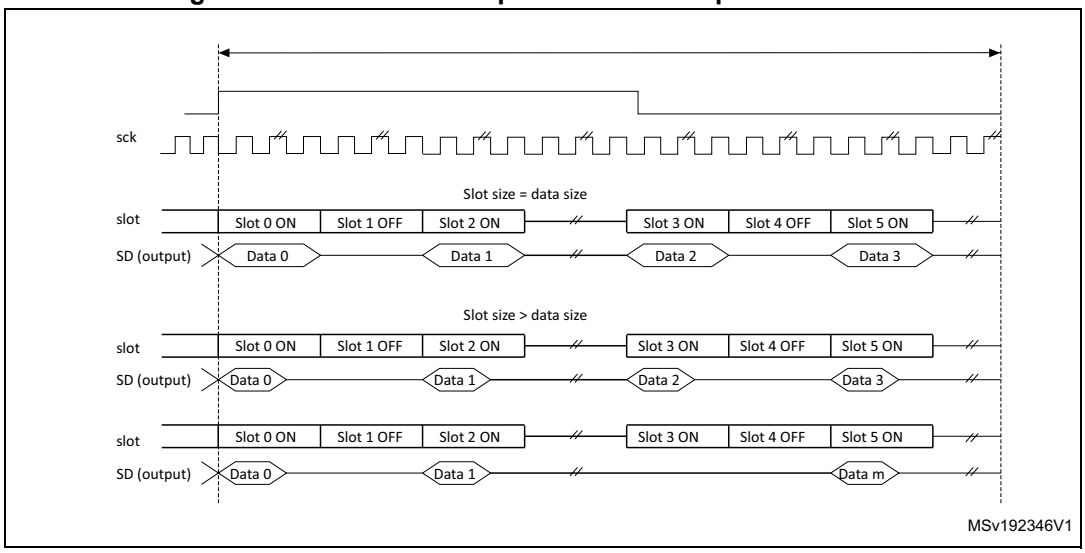
Figure 878 illustrates these behaviors.

Figure 878. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 879* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 879. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 878* and *Figure 879* are replaced by a drive with a value of 0.

70.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

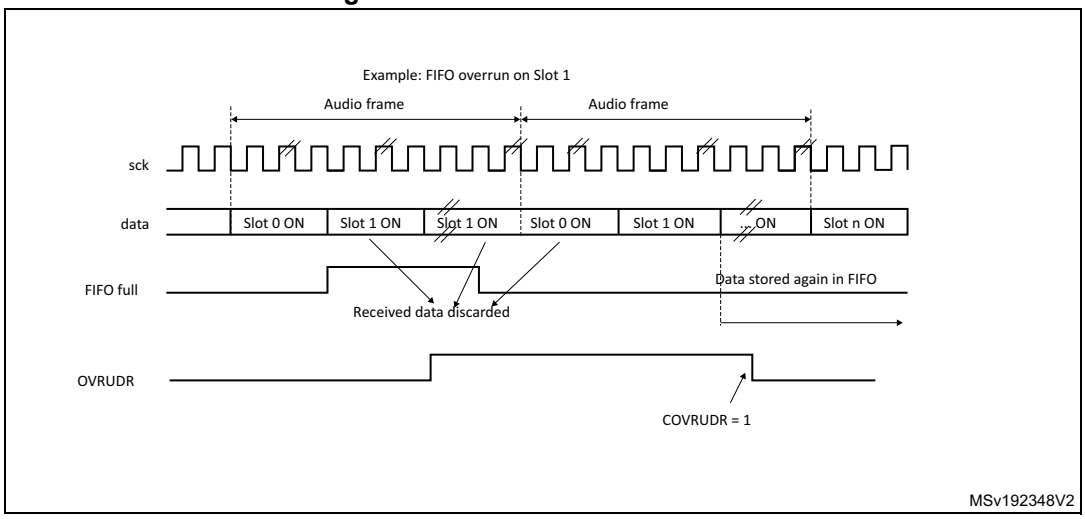
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 880*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 880. Overrun detection error



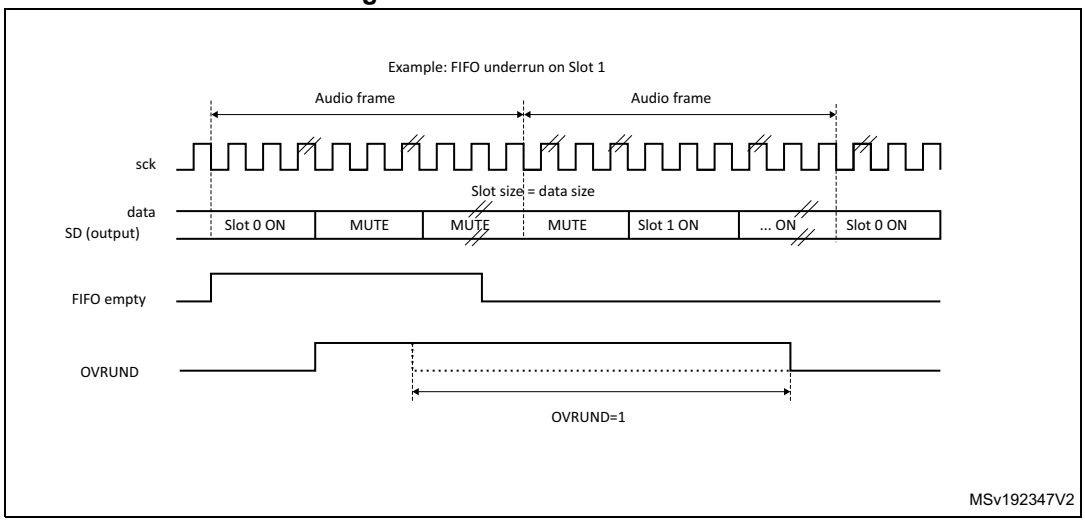
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 881*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 881. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

70.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

70.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 70.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

70.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 726*.

Table 726. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 70.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

70.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

70.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] must be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to *Table 718: External synchronization selection* for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

70.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 70.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

70.6.3 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

70.6.4 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

70.6.5 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

70.6.6 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

70.6.7 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 70.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

70.6.8 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

70.6.9 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

70.6.10 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, the master clock frequency is calculated according to the formula given in *Section 70.4.8*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

70.6.11 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

70.6.12 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

70.6.13 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

70.6.14 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

70.6.15 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 70.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

70.6.16 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

70.6.17 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

70.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 70.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 70.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 70.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

70.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 70.3: SAI implementation to check if it is available.

70.6.20 SAI hardware configuration register (SAI_HWCFGR)

Address offset: 0x03F0

Reset value: 0x0000 0108

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTION_REGOUT[7:4]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTION_REGOUT[3:0]				SPDIF_PDM[3:0]				FIFO_SIZE[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OPTION_REGOUT[7:0]:** Support of SAI_IOR register
0: SAI_IOR register is not implemented.
1: SAI_IOR register is implemented, with only 1 control bit at position 0.
2: SAI_IOR register is implemented, with 2 control bits at positions 0 and 1.
...
16: SAI_IOR register is implemented, with 16 control bits at positions 0 to 15.

Bits 11:8 **SPDIF_PDM[3:0]:** Support of SPDIF-OUT and PDM interfaces
0: SPDIF-OUT and PDM interfaces are not available.
1: SPDIF-OUT and PDM interfaces are available.

Bits 7:0 **FIFO_SIZE[7:0]:** FIFO size for SAIA and SAIB
4: the FIFO depth is 4 words.
8: the FIFO depth is 8 words.
12: the FIFO depth is 12 words.
16: the FIFO depth is 16 words.
20: the FIFO depth is 20 words.
24: the FIFO depth is 24 words.
28: the FIFO depth is 28 words.
32: the FIFO depth is 32 words.
others: Invalid values

70.6.21 SAI version register (SAI_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** SAI major revision

Bits 3:0 **MINREV[3:0]:** SAI minor revision

70.6.22 SAI identification register (SAI_IPIDR)

Address offset: 0x03F8

Reset value: 0x0013 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SAI identifier

70.6.23 SAI size identification register (SAI_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits returns the size of the memory region allocated to SAI registers.
The size of this memory region is of 1 Kbytes.

70.6.24 SAI register map

Table 727. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]		Res.	PRTCFG[1:0]		MODE[1:0]		
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]							MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0

Table 727. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x004C to 0x03EC	Reserved																																
0x03F0	SAI_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTION_REGOUT[7:0]								SPDIF_PDM[3:0]				FIFO_SIZE[7:0]							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0x03F4	SAI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value	0x0000 0022																															
0x03F8	SAI_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
0x03FC	SAI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

71 SPDIF receiver interface (SPDIFRX)

71.1 SPDIFRX interface introduction

The SPDIFRX interface handles S/PDIF audio protocol.

71.2 SPDIFRX main features

- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz[a] supported
- Supports audio IEC-60958 and IEC-61937, consumer applications
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

71.3 SPDIFRX functional description

The SPDIFRX peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. *Figure 882* shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[3:0] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB bus, and can handle two DMA channels:

- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

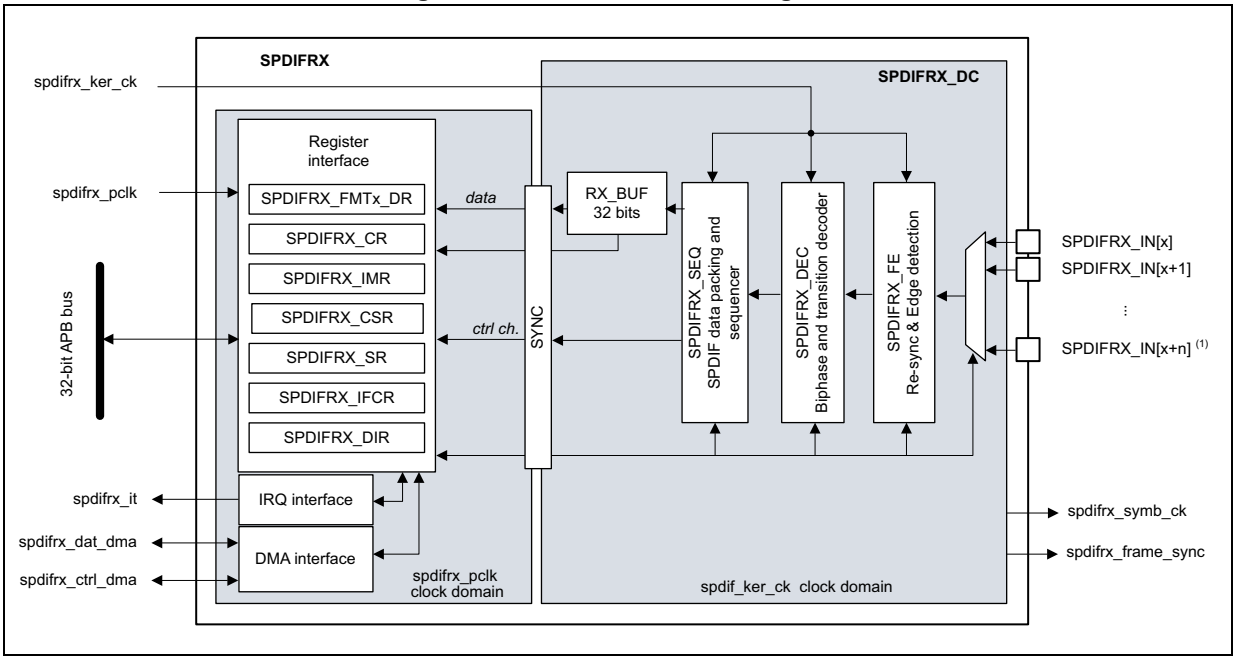
The SPDIFRX also offers a signal named spdifrx_frame_sync, which toggles every time that a sub-frame's preamble is detected. So the duty cycle is 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

a. Check the RCC capabilities in order to verify which sampling rates can be supported.

In addition the SPDIFRX also provides a signal named **spdifrx_symb_ck** toggling at the symbol rate.

Figure 882. SPDIFRX block diagram



1. 'n' is fixed to 3, and 'x' is set to 0.

71.3.1 SPDIFRX pins and internal signals

Table 728 lists the SPDIFRX internal input/output signals, *Table 729* the SPDIFRX pins (alternate functions).

Table 728. SPDIFRX internal input/output signals

Signal name	Signal type	Description
spdifrx_ker_ck	Digital input	SPDIFRX kernel clock
spdifrx_pclk	Digital input	SPDIFRX register interface clock
spdifrx_it	Digital output	SPDIFRX global interrupt
spdifrx_dat_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for data transfer
spdifrx_ctrl_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for channel status and user information transfer
spdifrx_frame_sync	Digital output	SPDIFRX frame rate synchronization signal
spdifrx_symb_ck	Digital output	SPDIFRX channel symbol clock

Table 729. SPDIFRX pins

Pin name	Pin type	Description
SPDIFRX_IN0	Digital input	Input 0 for S/PDIF signal
SPDIFRX_IN1	Digital input	Input 1 for S/PDIF signal

Table 729. SPDIFRX pins (continued)

Pin name	Pin type	Description
SPDIFRX_IN2	Digital input	Input 2 for S/PDIF signal
SPDIFRX_IN3	Digital input	Input 3 for S/PDIF signal

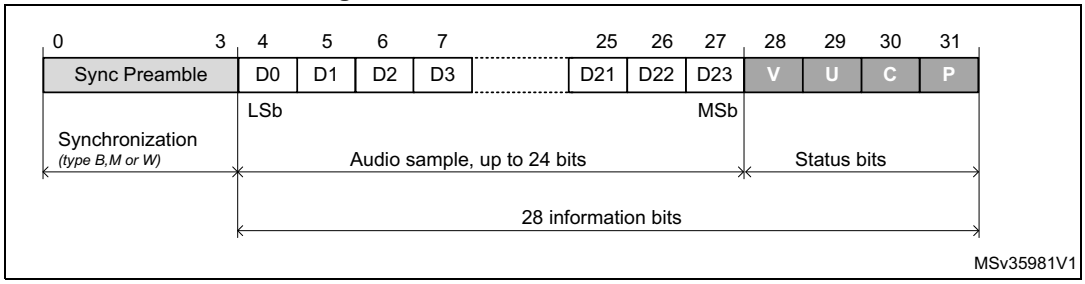
71.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block

A S/PDIF frame is composed of two sub-frames (see *Figure 883*). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit “V”) indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit “U”) carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit “C”) carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit “P”) carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

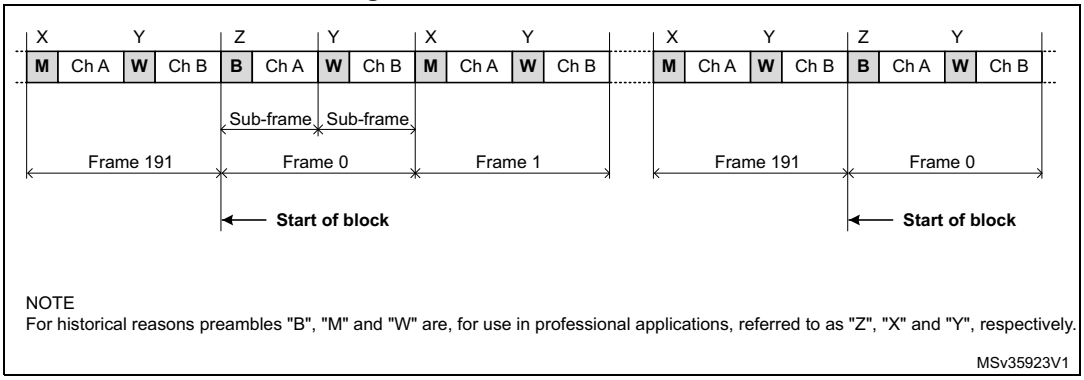
Figure 883. S/PDIF sub-frame format



For linear coded audio applications, the first sub-frame (left or “A” channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble “M”. However, the preamble changes to preamble “B” once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or “B” channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble “W”.

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.

Figure 884. S/PDIF block format



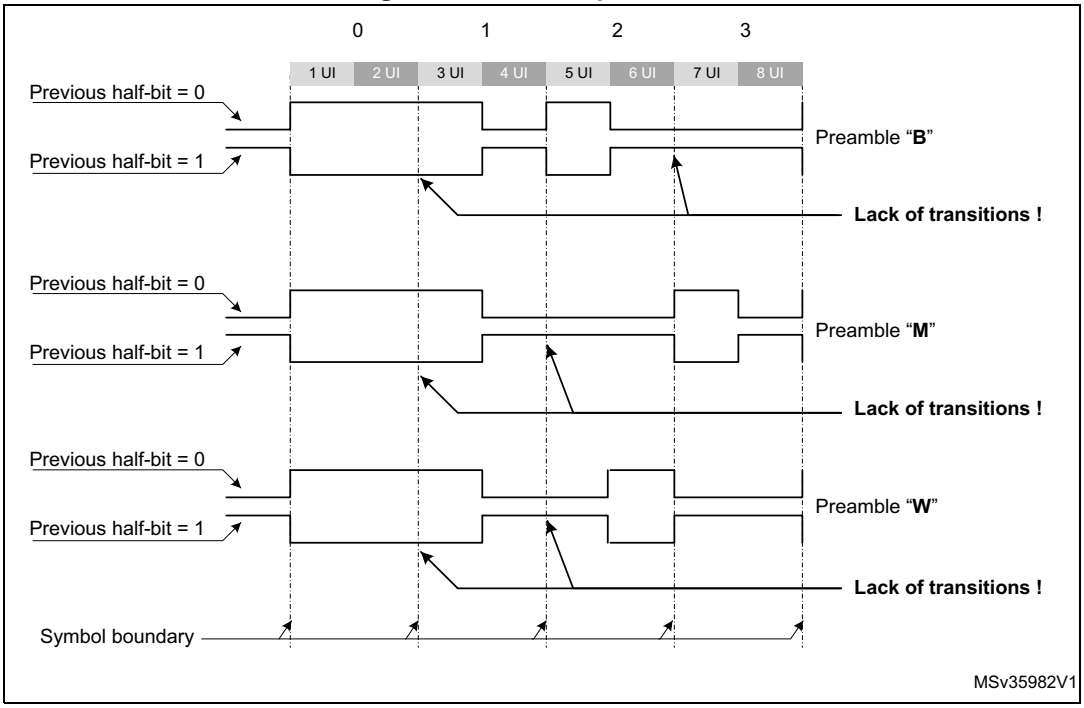
NOTE
For historical reasons preambles "B", "M" and "W" are, for use in professional applications, referred to as "Z", "X" and "Y", respectively.

MSv35923V1

Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the *Figure 885*.

Figure 885. S/PDIF preambles



MSv35982V1

Coding of information bits

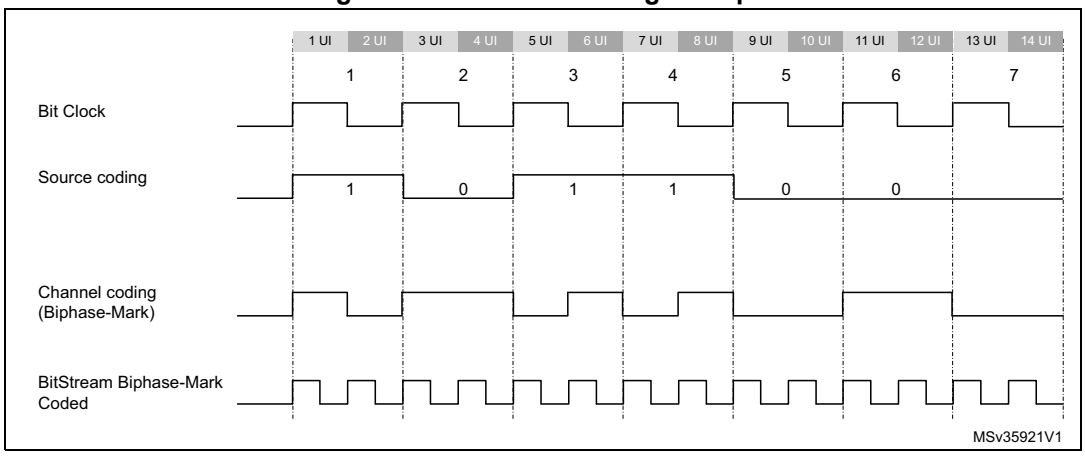
In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is

logical 0. However, it is different if the bit is logical 1. These states are named “UI” (unit interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

Figure 886. Channel coding example



71.3.3 SPDIFRX decoder (SPDIFRX_DC)

Main principle

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

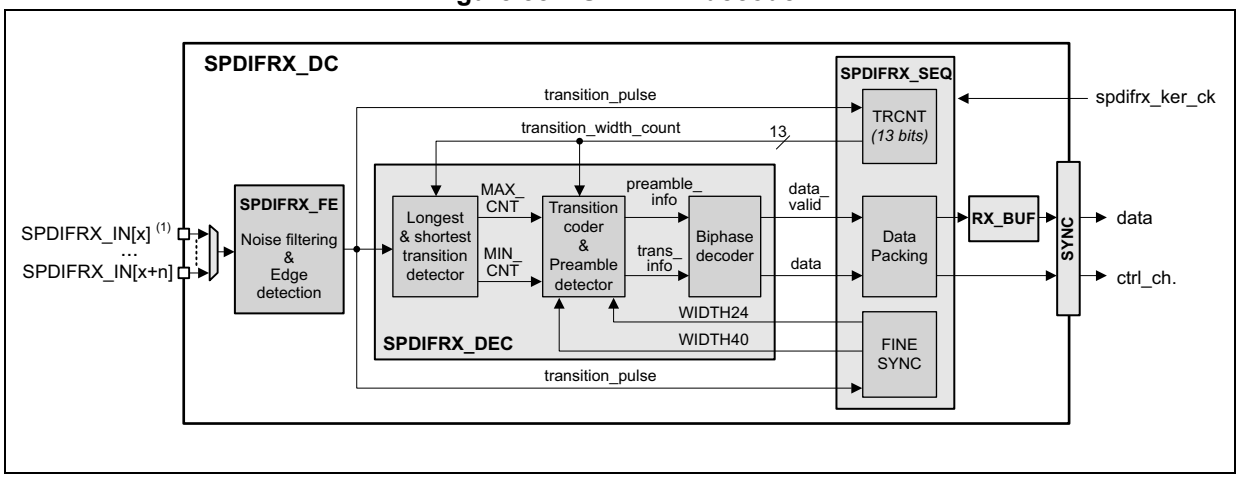
- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate

Figure 887 gives a detailed view of the SPDIFRX decoder.

Figure 887. SPDIFRX decoder



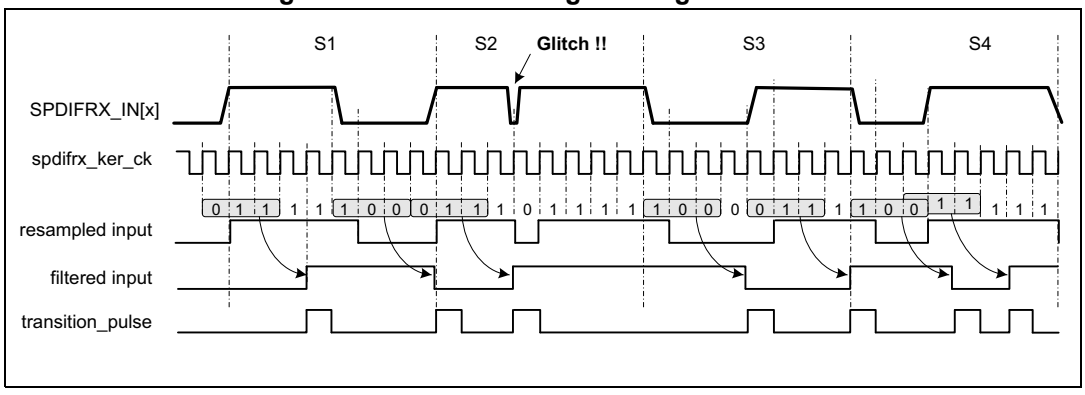
1. 'n' is fixed to 3, and 'x' is set to 0.

Noise filtering and rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

Figure 888. Noise filtering and edge detection



Longest and shortest transition detector

The **longest and shortest transition detector** block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.

The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.

The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

Transition coder and preamble detector

The **transition coder and preamble detector** block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see *Figure 887*). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH_{HI} and TH_{LO}.

- If the current transition width is less than (TH_{LO} - 1), then the data received is half part of data bit ‘1’, and is coded as TS.
- If the current transition width is greater than (TH_{LO} - 1), and less than TH_{HI}, then the data received is data bit ‘0’, and is coded as TM.
- If the current transition width is greater than TH_{HI}, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH_{HI} and TH_{LO} are elaborated using two different methods.

If the peripheral is doing its initial synchronization (‘coarse synchronization’), then the thresholds are computed as follow:

- TH_{LO} = MAX_CNT / 2.
- TH_{HI} = MIN_CNT + MAX_CNT / 2.

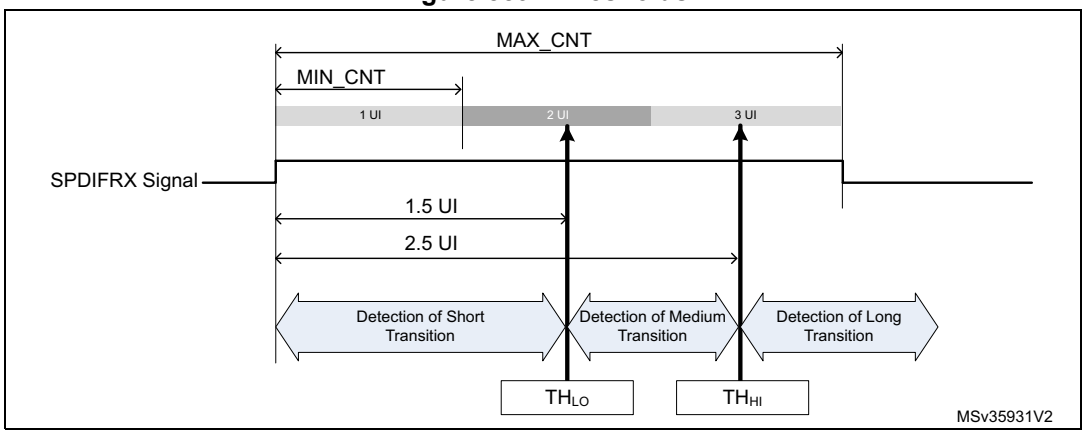
Once the ‘coarse synchronization’ is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH_{LO} and the length of 40 symbols (WIDTH40) for TH_{HI}. TH_{HI} and TH_{LO} are computed as follow:

- TH_{LO} = (WIDTH24) / 32
- TH_{HI} = (WIDTH40) / 32

This second synchronization phase is called the ‘fine synchronization’. Refer to *Figure 891* for additional information.

As shown in the figure hereafter, TH_{LO} is ideally equal to 1.5 UI, and to TH_{HI} 2.5 UI.

Figure 889. Thresholds



The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. *Table 730* shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder decode them.

Table 730. Transition sequence for preamble

Preamble type	Biphase data pattern	TRANS3	TRANS2	TRANS1	TRANS0
Preamble B	11101000	TL	TS	TS	TL
Preamble M	11100010	TL	TL	TS	TS
Preamble W	11100100	TL	TM	TS	TM

Bi-phase decoder

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the **transition coder and preamble detector** block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to *Section 71.3.9*, for additional information on error flags.

Data packing

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.

71.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception is. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

71.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. *Figure 890* shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the SPDIFRX switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See *Section 71.4* for additional information.
The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

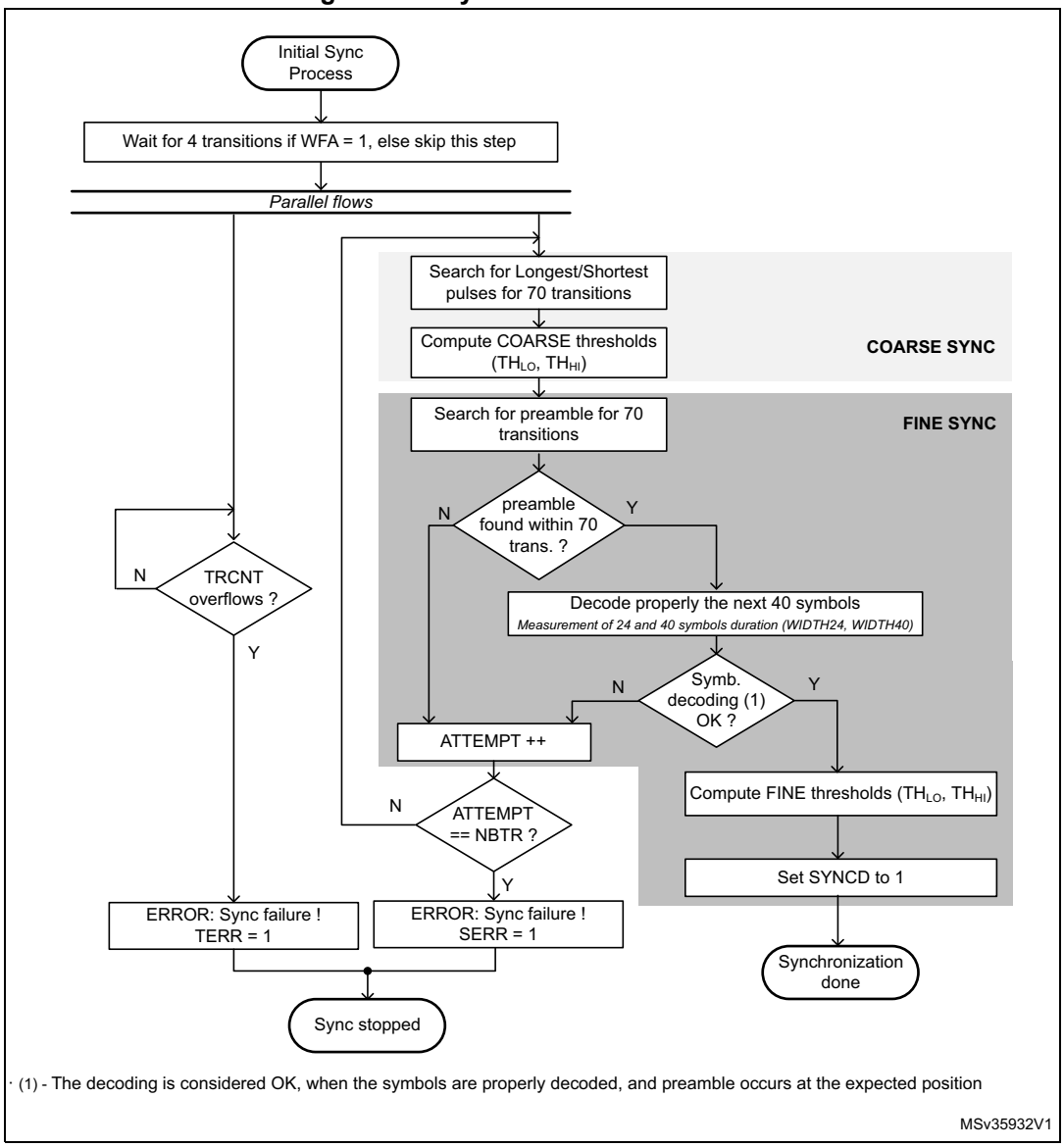
The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it can happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag.
When the SPDIFRX is able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYNCD is set to 1. Refer to *Transition coder and preamble detector* for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) starts when the next “B” preamble is detected (see *Figure 894*).Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11. Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.

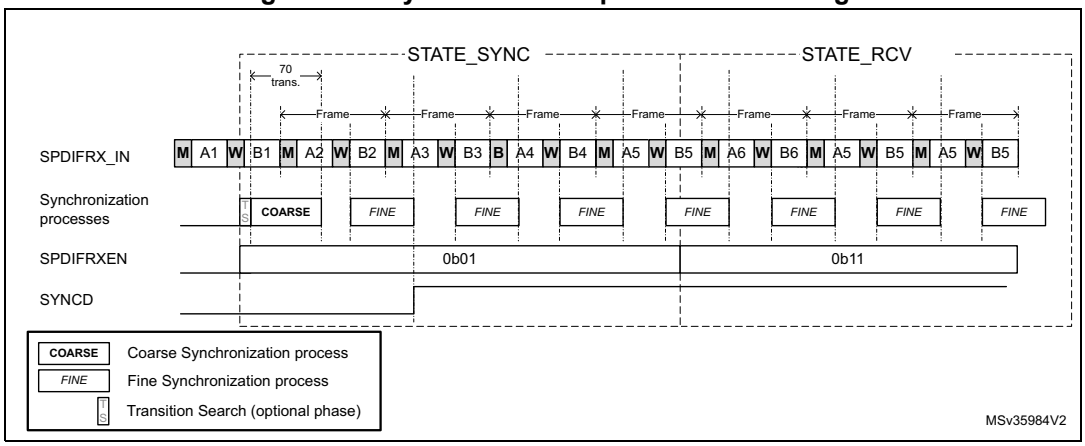
Figure 890. Synchronization flowchart



Refer to *Frame structure and synchronization error* for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in *Figure 891* in order to continuously track S/PDIF synchronization.

Figure 891. Synchronization process scheduling



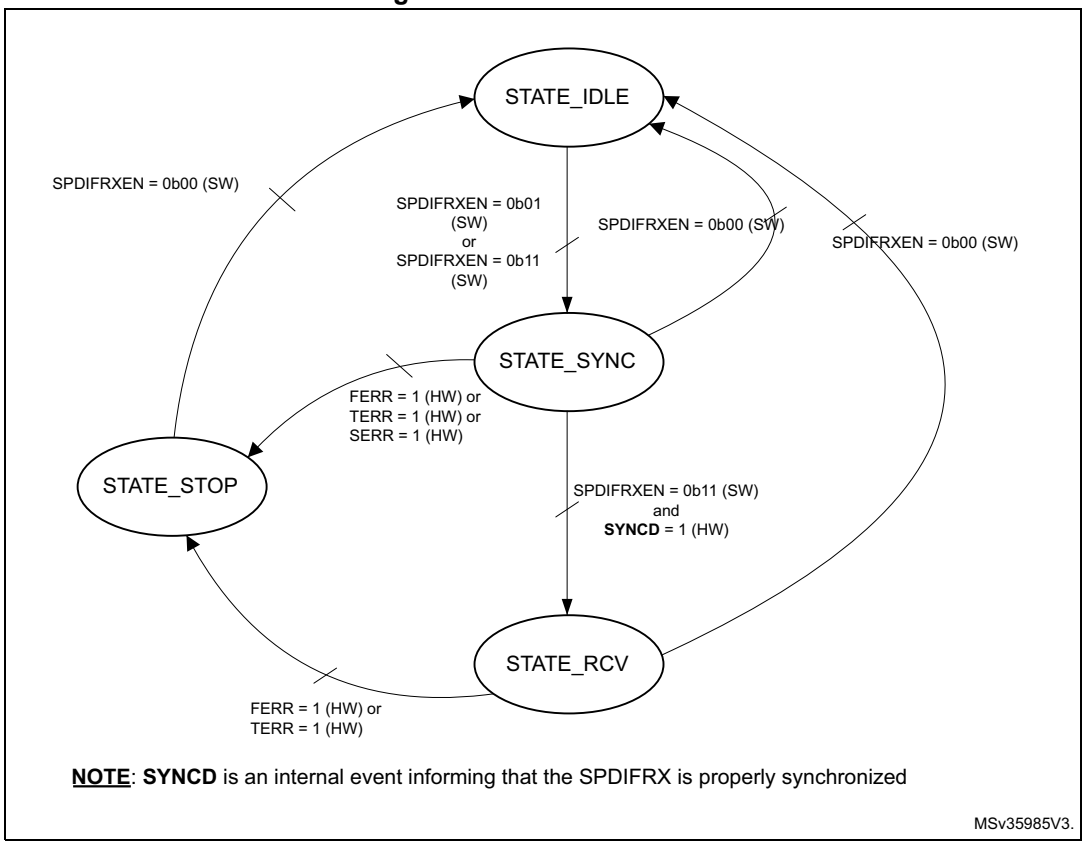
71.3.6 SPDIFRX handling

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- STATE_IDLE:
  The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.
- STATE_SYNC:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.
- STATE_RCV:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.
- STOP_STATE:
  The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

Figure 892 shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under SPDIFRX control are followed by the mention “(HW)”.

Figure 892. SPDIFRX states



When SPDIFRX is in STATE_IDLE:

- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11

When SPDIFRX is in STATE_SYNC:

- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1

When SPDIFRX is in STATE_RCV:

- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.

When SPDIFRX is in STATE_STOP:

- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.

When SPDIFRXEN is set to 0, the SPDIFRX is disabled, meaning that all the state machines are reset, and RX_BUF is flushed. Note as well that flags FERR, SERR and TERR are reset.

71.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF are immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF are transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word is completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is T_{SAMPLE} - $2T_{PCLK}$ - $2T_{spdifrx_ker_ck}$

Where T_{SAMPLE} is the audio sampling rate of the received stereo audio samples, T_{PCLK} is the period of spdifrx_pclk clock, and $T_{spdifrx_ker_ck}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to *Section 71.3.12: DMA interface* for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see *Mixing data and control flow*).

Note that PE bit stands for parity rrror bit, and is set to 1 when a parity error is detected in the decoded sub-frame.
The PT field carries the preamble type (B, M or W).

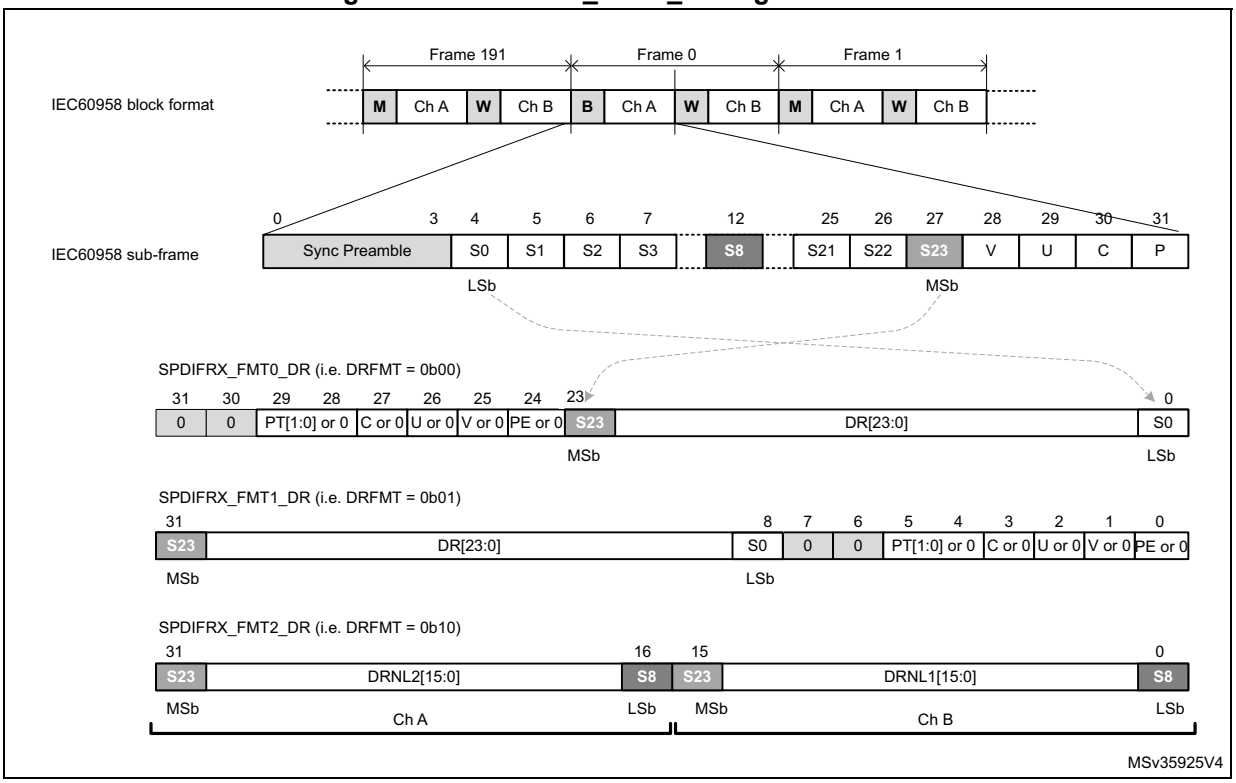
V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in *Figure 893*.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] (x = 2 to 0), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR"

Figure 893. SPDIFRX_FMTx_DR register format



Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT= 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA are always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see *Figure 895*).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see *Section 71.3.8*)

Mixing data and control flow

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field must be kept into the data register (SPDIFRX_FMTx_DR).

- When bit PMSK = 1, the parity error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

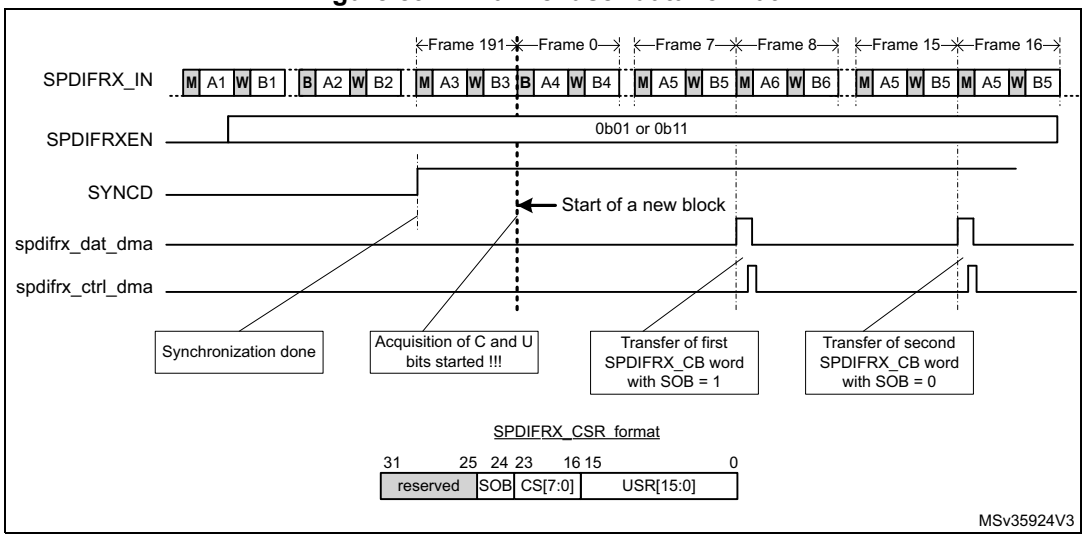
- When bit CUMSK = 1, the channel status, and used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the preamble type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

71.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition starts at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits are received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see *Figure 894*).

If CS[0] corresponds to the first bit of a new block, the bit SOB is set to 1. Refer to *Section 71.5.8*. A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

Figure 894. Channel/user data format



Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.

71.3.9 Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:

- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the SPDIFRX

Refer to *Figure 892* for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

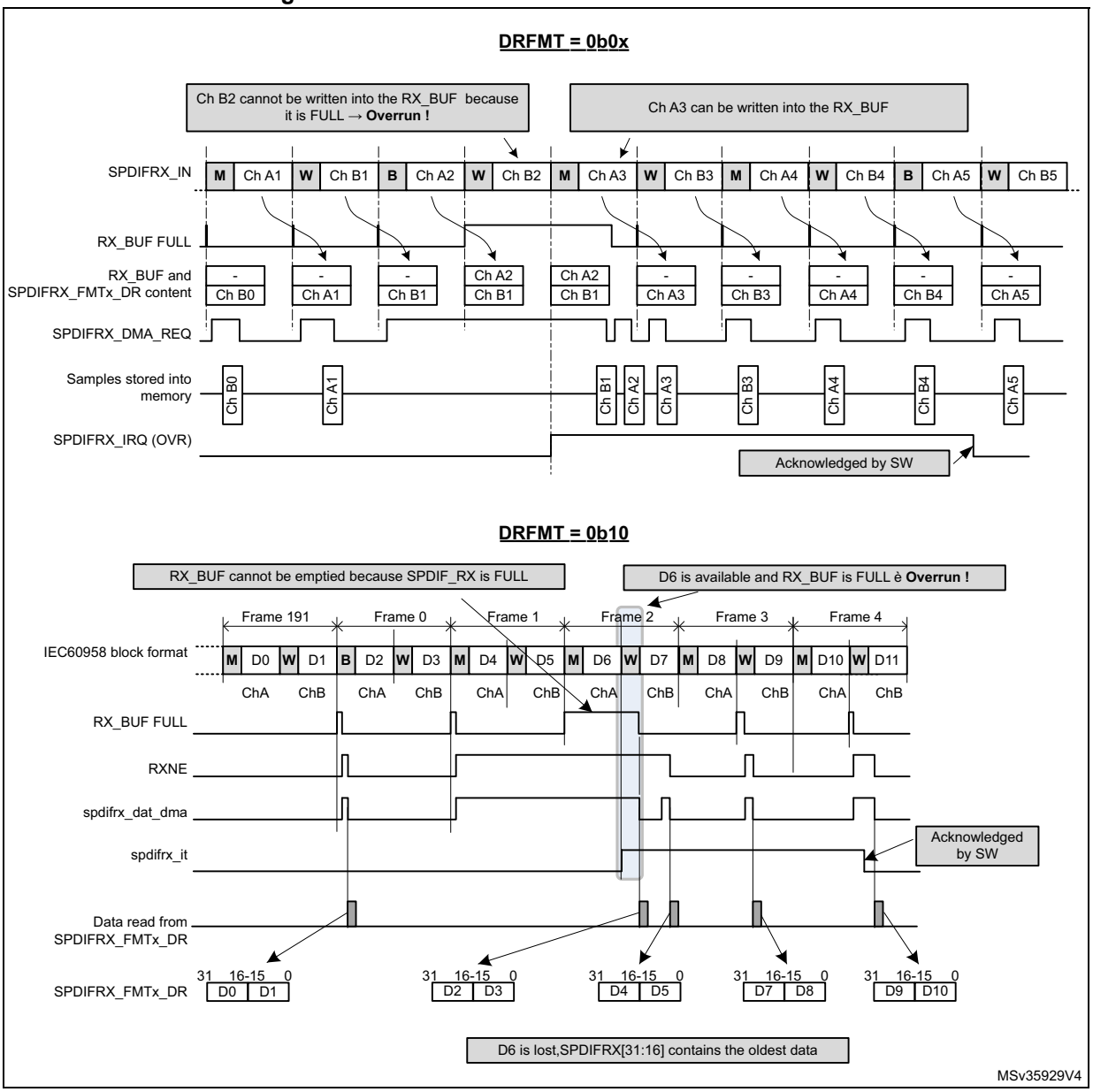
If the software wants to guarantee the coherency between the data read in the SPDIFRX_FMTx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered. The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the SPDIFRX stores the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See *Figure 895*.

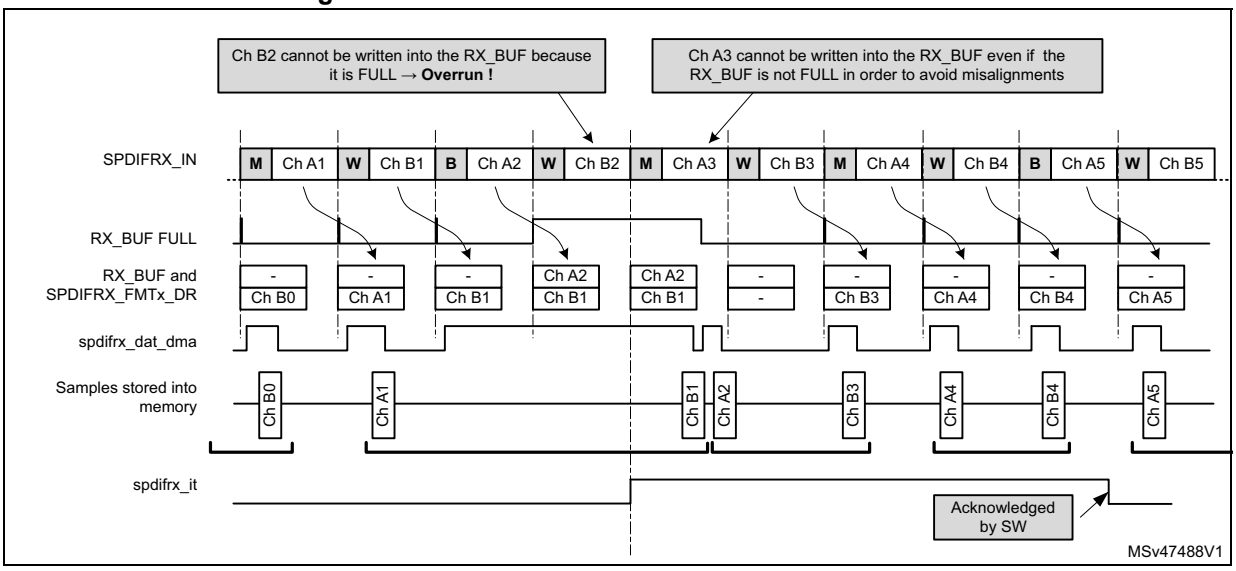
Figure 895. S/PDIF overrun error when RXSTEO = 0



If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to *Figure 896*.

The OVR flag is cleared by software, by setting the OVRCF bit to 1.

Figure 896. S/PDIF overrun error when RXSTEO = 1



71.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on *Figure 882*).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC must re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate to up to 12.288 MHz, the sample rate must be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

Table 731. Minimum spdifrx_ker_ck frequency versus audio sampling rate(1)

Symbol rate	Minimum spdifrx_ker_ck frequency	Comments
3.072 MHz	33.8 MHz	For 48 kHz stream
6.144 MHz	67.6 MHz	For 96 kHz stream
12.288 MHz	135.2 MHz	For 192 kHz stream

1. Check the RCC capabilities in order to verify which sampling rates can be supported.

71.3.11 Symbol clock generation

The SPDIFRX block provides a symbol clock on signal named **spdifrx_symb_ck**, which can be used as the reference kernel clock for another audio device such as SAI or SPI/I2S. It can be used for SPDIFRX to I2S bridge function.

The symbol clock is built using the values of WIDTH24, WIDTH40 and the symbol boundaries.

- During the reception of the sub-frame sync preambles, the falling and rising edges of the symbol clock are built from the WIDTH24 and WIDTH40 values. Note that WIDTH24 and WIDTH40 are also used for the generation of the symbol clock, when the SPDIFRX is STATE_STOP or STATE_IDLE. See *Table 732* for details.
- During the reception of the sub-frame payload, the SPDIFRX uses the symbols boundaries to generate the rising edge, the WIDTH24 and WIDTH40 values for the generation of the falling edge.

The duty cycle of the symbol clock is close to 50% during the reception of the sub-frame payload. However, the duty cycle can be altered when the SPDIFRX transitions from a symbol clock generated with WIDTH24 and WIDTH40 to a clock generated by the symbol clock boundaries or vice-versa.

The symbol clock has an important jitter mainly due to:

- The re-sampling of the S/PDIF signal with spdifrx_ker_ck clock
- The transition of the symbol clock generation mode

For that reason the application must consider the quality degradation if the symbol clock is used as the reference clock for A/D or D/A converters.

The generation of this symbol clock is controlled by the CKSEN bit. When CKSEN = ‘1’, the clock symbol is generated when the SPDIFRX completes successfully the first fine synchronization (SYNCD = 1), and when it receives correct data from the selected SPDIFRX input.

When the SPDIFRX goes to STATE_STOP, or STATE_IDLE, the symbol clock is gated if the bit CKSBKPEN = ‘0’. If the CKSBKPEN = ‘1’, then a backup symbol clock is still generated if the SPDIFRX is properly synchronized (i.e. valid values available for WIDTH24 and WIDTH40). *Table 732* gives more details on the conditions controlling the generation of the symbol clock.

Table 732. Conditions of spdifrx_symb_ck generation

SPDIFRX states and conditions	CKSEN	CKSBKPEN	spdifrx_symb_ck state
Any state	0	X	Disabled
– SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or, – SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input.	1	0	Enabled
– SPDIFRX in STATE_IDLE or, – SPDIFRX in STATE_STOP or, – SPDIFRX did not complete the fine synchronization (on-going) – SPDIFRX is in STATE_RCV, but no data (transitions) detected on the selected SPDIFRX input.		0	Disabled

Table 732. Conditions of spdifrx_symb_ck generation (continued)

SPDIFRX states and conditions	CKSEN	CKSBKPEN	spdifrx_symb_ck state
– SPDIFRX in STATE_IDLE, but with valid values for WIDTH40 and WIDTH24 or – SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or, – SPDIFRX in STATE_SYNC the on-going fine synchronization is not completed, but WIDTH40 and WIDTH24 contain the valid values from the previous synchronization or, – SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input or, – SPDIFRX in STATE_STOP, but with valid values for WIDTH40 and WIDTH24.	1	1	Enabled
– SPDIFRX in IDLE, with invalid values for WIDTH40 and WIDTH24 or, – SPDIFRX in STOP with invalid values for WIDTH40 and WIDTH24 (SERR = '1') or, – SPDIFRX in STATE_SYNC with invalid values for WIDTH40 and WIDTH24, and did not completed the on-going fine synchronization or, – SPDIFRX in STATE_RCV and no transitions detected on the selected SPDIFRX input			Disabled

Note that when the flag SERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization.

Note that when both CKSEN and CKSBKPEN are set to ‘1’, the symbol clock looses some transitions when the SPDIFRX switches from STATE_SYNC or STATE_RCV to STATE_STOP, or STATE_IDLE.

The bits CKSEN and CKSBKPEN are located into *SPDIFRX control register (SPDIFRX_CR)*.

71.3.12 DMA interface

The SPDIFRX interface is able to perform communication using the DMA.

Note: The user must refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:

- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

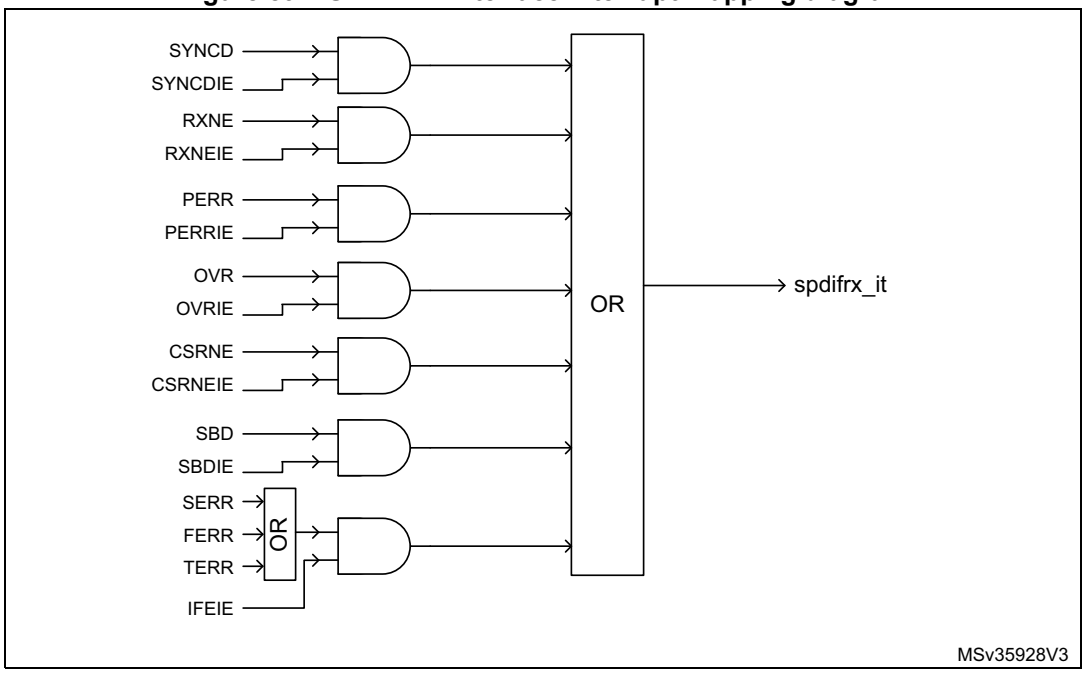
For the use of DMA for the control data refer to *Section 71.3.8*.

71.3.13 Interrupt generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

Figure 897. SPDIFRX interface interrupt mapping diagram



Clearing interrupt source

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

Note: The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).
The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.

71.3.14 Register protection

The SPDIFRX block embeds some hardware protection avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

Table 733. Bit field property versus SPDIFRX state

Registers	Field	SPDIFRXEN		
		00 (STATE_IDLE)	01 (STATE_SYNC)	11 (STATE_RCV)
SPDIFRX_CR	INSEL	rw	r	r
	WFA	rw	r	r
	NBTR	rw	r	r
	CHSEL	rw	r	r
	CBDMAEN	rw	rw	rw
	PTMSK	rw	rw	rw
	CUMSK	rw	rw	rw
	VMSK	rw	rw	rw
	PMSK	rw	rw	rw
	DRFMT	rw	rw	r
	RXSTEO	rw	rw	r
	RXDMAEN	rw	rw	rw
SPDIFRX_IMR	All fields	rw	rw	rw

The table clearly shows that fields such as INSEL must be programmed when the SPDIFRX is in STATE_IDLE. In the others SPDIFRX states, the hardware prevents writing to this field.

Note: Even if the hardware allows the writing of CBDMAEN and RXDMAEN "on-the-fly", it is not recommended to enable the DMA when the SPDIFRX already receives data.

Each of the mask bits (such as PMSK, VMSK) can be changed "on-the-fly" at any SPDIFRX state, but any change does not affect data which are already hold in SPDIFRX_FMTx_DR.

71.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status and user information both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition

A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE_SYNC, with bit WFA set to 1. The description hereafter focuses on detection. It is also possible to implement this function as follows:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

  For those two implementations, the bit WFA is set to 0.

71.4.1 Initialization phase

- The initialization function looks like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to *Table 731* in order to define the minimum clock frequency versus supported audio sampling rate.
    Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to handle: for example if the application is able to handle streams to up to 96 kHz, then $F_{spdifrx_ker_ck}$ must be at least 704 x 96 kHz = 67.6 MHz
- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL must select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status are read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU receives interrupts coming either from DMA or SPDIFRX.

71.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization is properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRX_IFCR register to 1.
- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries are reached.
  - FERR indicates that the reading of information after synchronization fails (such as unexpected preamble, bad data decoding).
- If PERR is set to 1, it means that a parity error is detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action can be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRX_IFCR register to 1.

71.4.3 Handling of interrupts coming from DMA

If an interrupt comes from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (that is PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The SOB bit, when set to 1 indicates the start of a new block. This information helps the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word always corresponds to the start of a new block.

If an interrupt comes from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.

71.5 SPDIFRX interface registers

71.5.1 SPDIFRX control register (SPDIFRX_CR)

Only 32-bit accesses are allowed in this register.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CKSBKPEN**: Backup symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a backup symbol clock if CKSEN = 1.
0: The SPDIFRX does not generate a backup symbol clock.

Bit 20 **CKSEN**: Symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a symbol clock.
0: The SPDIFRX does not generate a symbol clock.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **INSEL[2:0]**: SPDIFRX input selection(1)
000: SPDIFRX_IN0 selected
001: SPDIFRX_IN1 selected
010: SPDIFRX_IN2 selected
011: SPDIFRX_IN3 selected
other: reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WFA**: Wait for activity(1)
This bit is set/reset by software.
1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization.
0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization.

Bits 13:12 **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase(1)
00: No re-try is allowed (only one attempt)
01: 3 re-tries allowed
10: 15 re-tries allowed
11: 63 re-tries allowed

Bit 11 **CHSEL**: Channel selection(1)
This bit is set/reset by software.
1: The control flow takes the channel status from channel B.
0: The control flow takes the channel status from channel A.

Bit 10 **CBDMAEN**: Control buffer DMA enable for control flow(1)
This bit is set/reset by software.
1: DMA mode is enabled for reception of channel status and used data information.
0: DMA mode is disabled for reception of channel status and used data information.
Note: When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9 **PTMSK**: Mask of preamble type bits(1)
This bit is set/reset by software.
1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The preamble type bits are copied into the SPDIFRX_FMTx_DR.

Bit 8 **CUMSK**: Mask of channel status and user bits(1)
This bit is set/reset by software.
1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR.

Bit 7 **VMSK**: Mask of validity bit(1)
This bit is set/reset by software.
1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The validity bit is copied into the SPDIFRX_FMTx_DR.

Bit 6 **PMSK**: Mask parity error bit(1)
This bit is set/reset by software.
1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The parity error bit is copied into the SPDIFRX_FMTx_DR.

Bits 5:4 **DRFMT[1:0]**: RX data format(1)
This bit is set/reset by software.
11: reserved
10: Data sample are packed by setting two 16-bit sample into a 32-bit word.
01: Data samples are aligned in the left (MSB)
00: Data samples are aligned in the right (LSB).

Bit 3 **RXSTEO**: Stereo mode(1)

This bit is set/reset by software.

1: The peripheral is in stereo mode.

0: The peripheral is in mono mode.

Note: This bit is used in case of overrun situation in order to handle misalignment.

Bit 2 **RXDMAEN**: Receiver DMA enable for data flow(1)

This bit is set/reset by software.

1: DMA mode is enabled for reception.

0: DMA mode is disabled for reception.

Note: When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0 **SPDIFRXEN[1:0]**: Peripheral block enable(1)

This field is modified by software.

It must be used to change the peripheral phase among the three possible states: STATE_IDLE, STATE_SYNC and STATE_RCV.

00: Disable SPDIFRX (STATE_IDLE).

01: Enable SPDIFRX synchronization only.

10: Reserved

11: Enable SPDIF receiver.

Note: It is not possible to transition from STATE_RCV to STATE_SYNC, the user must first go the STATE_IDLE.

it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is performed goes to STATE_RCV.

1. Refer to *Section 71.3.14: Register protection* for additional information on fields properties.

71.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFE IE	SYNCD IE	SBLK IE	OVR IE	PERR IE	CSRNE IE	RXNE IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **IFEIE**: Serial interface error interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SERR = 1, TERR = 1, or FERR = 1 in the SPDIFRX_SR register.

Bit 5 **SYNCDIE**: Synchronization done

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SYNCD = 1 in the SPDIFRX_SR register.

Bit 4 **SBLKIE**: Synchronization block detected interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever SBD = 1 in the SPDIFRX_SR register.

Bit 3 **OVRIE**: Overrun error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever OVR=1 in the SPDIFRX_SR register.

Bit 2 **PERRIE**: Parity error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever PERR=1 in the SPDIFRX_SR register.

Bit 1 **CSRNEIE:** Control buffer ready interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever CSRNE = 1 in the SPDIFRX_SR register.

Bit 0 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever RXNE=1 in the SPDIFRX_SR register.

71.5.3 SPDIFRX status register (SPDIFRX_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WIDTH5[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
							r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **WIDTH5[14:0]**: duration of 5 symbols counted with spdifrx_ker_ck
This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck.
For example if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is:
Fs = 5 x $F_{spdifrx_ker_ck}$ / (WIDTH5 x 64) ~ 44.6 kHz, so the closest standard sampling rate is 44.1 kHz.
Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TERR**: Time-out error

This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No sequence error is detected.

1: Sequence error is detected.

Bit 7 **SERR**: Synchronization error

This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE = 1 in the SPDIFRX_IMR register.

0: No synchronization error is detected.

1: Synchronization error is detected.

Bit 6 **FERR**: Framing error

This bit is set by hardware when an error occurs during data reception: such as preamble not at the expected place, short transition not grouped by pairs.

This is set by the hardware only if the synchronization is completed (SYNCD = 1).

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No Manchester violation detected

1: Manchester violation detected

Bit 5 **SYNCD**: Synchronization done

This bit is set by hardware when the initial synchronization phase is properly completed.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SYNCDIE = 1 in the SPDIFRX_IMR register.

0: Synchronization is pending.

1: Synchronization is completed.

Bit 4 **SBD**: Synchronization block detected

This bit is set by hardware when a “B” preamble is detected.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register.

0: No “B” preamble is detected.

1: “B” preamble is detected.

Bit 3 **OVR**: Overrun error

This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.

0: No overrun error

1: Overrun error is detected.

Note: When this bit is set, the SPDIFRX_FMTx_DR register content is not lost but the last data received are.

Bit 2 **PERR**: Parity error
This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.
0: No parity error
1: Parity error

Bit 1 **CSRNE**: Control buffer register not empty
This bit is set by hardware when a valid control information is ready.
This flag is cleared when reading SPDIFRX_CSR register.
An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register.
0: No control word available on SPDIFRX_CSR register
1: A control word is available on SPDIFRX_CSR register.

Bit 0 **RXNE**: Read data register not empty
This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.
This flag is cleared by reading the SPDIFRX_FMTx_DR register.
An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.
0: Data is not received.
1: Received data is ready to be read.

71.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
										w	w	w	w		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SYNCDCF**: clears the synchronization done flag
Writing 1 in this bit clears the flag SYNCD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 4 **SBDCF**: clears the synchronization block detected flag
Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 3 **OVRCF**: clears the overrun error flag
Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 2 **PERRCF**: clears the parity error flag
Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bits 1:0 Reserved, must be kept at reset value.

71.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 00:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:16]							
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: Preamble B received
10: Preamble M received
11: Preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 27 **C**: channel status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

71.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 01:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[7:0]								Res.	Res.	PT[1:0]		C	U	V	PE
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:8 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: preamble B received
10: preamble M received
11: preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 3 **C**: channel Status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 2 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 1 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 0 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

71.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.

The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRNL2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRNL1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DRNL2[15:0]**: data value
This field contains the channel A

Bits 15:0 **DRNL1[15:0]**: data value
This field contains the channel B

71.5.8 SPDIFRX channel status register (SPDIFRX_CSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]							
							r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SOB**: start of block
This bit indicates if the bit CS[0] corresponds to the first bit of a new block
0: CS[0] is not the first bit of a new block
1: CS[0] is the first bit of a new block

Bits 23:16 **CS[7:0]**: channel A status information
Bit CS[0] is the oldest value

Bits 15:0 **USR[15:0]**: user data information
Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B.
So USR[n] bits come from channel A is n is even, otherwise they come from channel B.

71.5.9 SPDIFRX debug information register (SPDIFRX_DIR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TLO[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	THI[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **TLO[12:0]**: threshold LOW (TLO = 1.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x TLO x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/3
Note that TLO is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **THI[12:0]**: threshold HIGH (THI = 2.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x THI x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/5
Note that THI is updated by the hardware when SYNCD goes high, and then every frame.

71.5.10 SPDIFRX version register (SPDIFRX_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** SPDIFRX major revision
Major revision is 1.

Bits 3:0 **MINREV[3:0]:** SPDIFRX minor revision
Minor revision is 2.

71.5.11 SPDIFRX identification register (SPDIFRX_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SPDIFRX identifier

71.5.12 SPDIFRX size identification register (SPDIFRX_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits return the size of the memory region allocated to SPDIFRX registers. The size of this memory region is of 1 Kbyte.

71.5.13 SPDIFRX interface register map

Table 734. SPDIFRX interface register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPDIFRX_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]			Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	Reset value											0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPDIFRX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEIE	SYNCDIE	SBLKIE	OVRIE	PERRIE	CSRNEIE	RXNEIE
	Reset value																										0	0	0	0	0	0	0
0x08	SPDIFRX_SR	Res.	WIDTH5[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C	SPDIFRX_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
	Reset value																											0	0	0	0		
0x10	SPDIFRX_FMT0_DR	Res.	Res.	PT[1:0]		C	U	V	P E	DR[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_FMT1_DR	DR[23:0]																Res.		PT[1:0]		C	U	V	P E								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_FMT2_DR	DRNL2[15:0]																DRNL1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 734. SPDIFRX interface register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	SPDIFRX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]								USR[15:0]															
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPDIFRX_DIR	Res.	Res.	Res.	TLO[12:0]													Res.	Res.	Res.	THI[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x3F4	SPDIFRX_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0x3F8	SPDIFRX_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	SPDIFRX_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

72 Controller area network with flexible data rate (FDCAN)

72.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 898*) consists of two CAN modules, a shared message RAM, and a clock calibration unit. Refer to the product memory organization for the base address of each of them.

FDCAN modules are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

In addition, the first CAN module FDCAN1 supports time triggered CAN (TTCAN), specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The FDCAN1 contains additional registers, specific to the time triggered feature. The CAN FD option can be used together with event-triggered and time triggered CAN communication.

A 10-Kbyte message RAM implements filters, receive FIFOs, receive buffers, transmit event FIFOs, transmit buffers (and triggers for TTCAN). This memory is shared between the FDCAN modules.

The common clock calibration unit is optional. It can be used to generate a calibrated clock for each FDCAN from the HSI internal RC oscillator and the PLL, by evaluating CAN messages received by the FDCAN1.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 735* and *Table 736*.

Table 735. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_cal_it	Digital output	FDCAN calibration interrupt
fdcan1_intr0_it	Digital output	FDCAN1 interrupt0
fdcan1_intr1_it		FDCAN1 interrupt1
fdcan2_intr0_it	Digital output	FDCAN2 interrupt0
fdcan2_intr1_it		FDCAN2 interrupt1
fdcan1_swt[0:5]	Digital input	Stop watch trigger input
fdcan1_evt[0:5]		Event trigger input
fdcan1_ts[0:15]		External timestamp vector
fdcan1_soc	Digital output	Start of cycle pulse
fdcan1_rtp		Register time mark pulse
fdcan1_tmp		Trigger time mark pulse

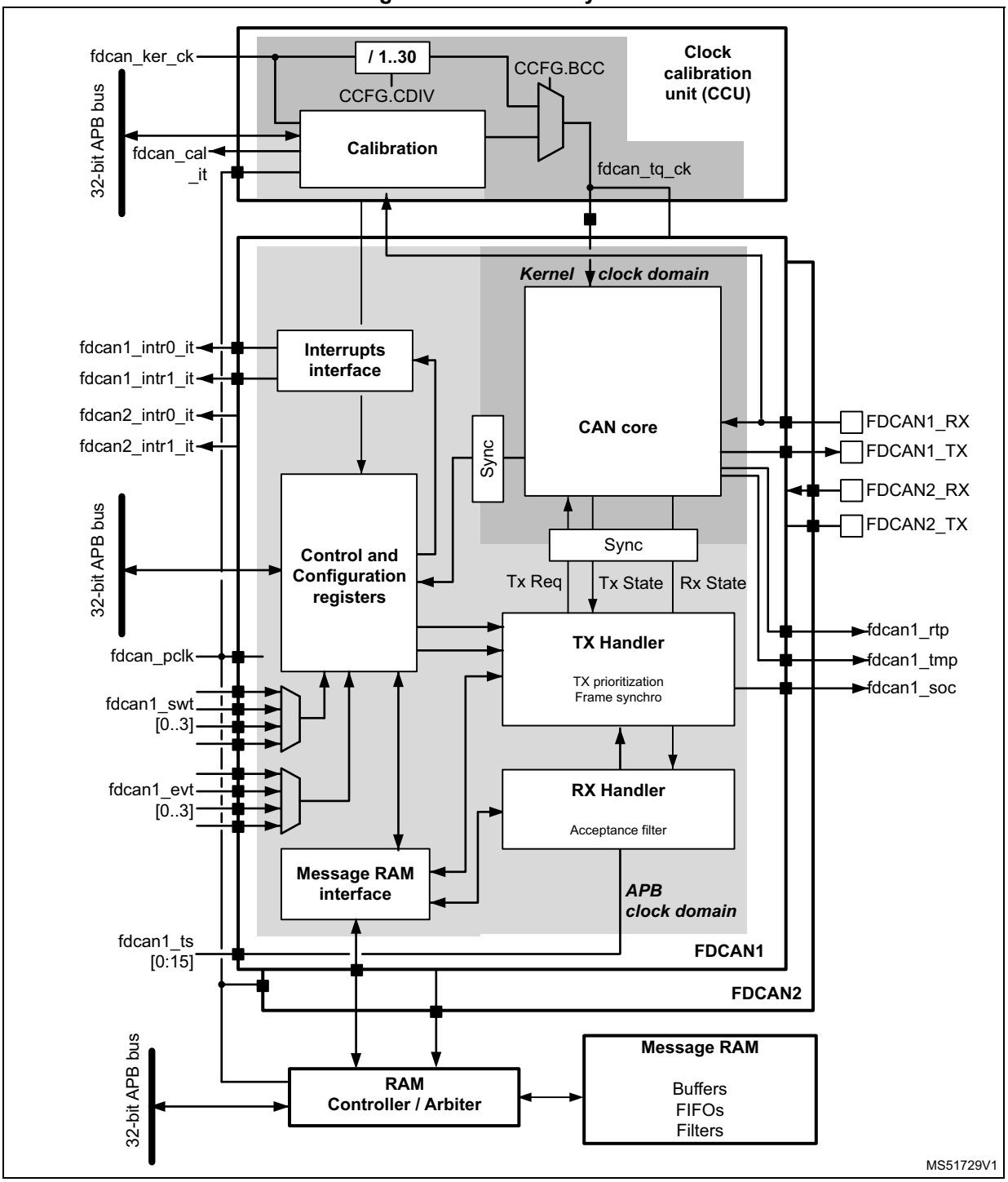
Table 736. CAN subsystem I/O pins

Name	Type	Description
FDCAN1_RX	Digital input	FDCAN1 receive pin
FDCAN1_TX	Digital output	FDCAN1 transmit pin
FDCAN2_RX	Digital input	FDCAN2 receive pin
FDCAN2_TX	Digital output	FDCAN2 transmit pin

Table 737. CAN triggers

Type	ID	Source	Destination
Internal	SWT2	ETH1_PPS	-
	SWT3	ETH2_PPS	-
	EWT2	ETH1_PPS	-
	EWT3	ETH2_PPS	-
External	fdcan_tt_soc	-	TIM5_ITR15
	fdcan_tt_tmp	-	TIM5_TI1_1
	fdcan_tt_rtp	-	TIM5_TI1_2

Figure 898. CAN subsystem



72.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015, -4
- CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware (FDCAN1 only)
- Event synchronized time-triggered communication supported (FDCAN1 only)
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two configurable receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO / queue
- Configurable transmit event FIFO
- FDCAN modules share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

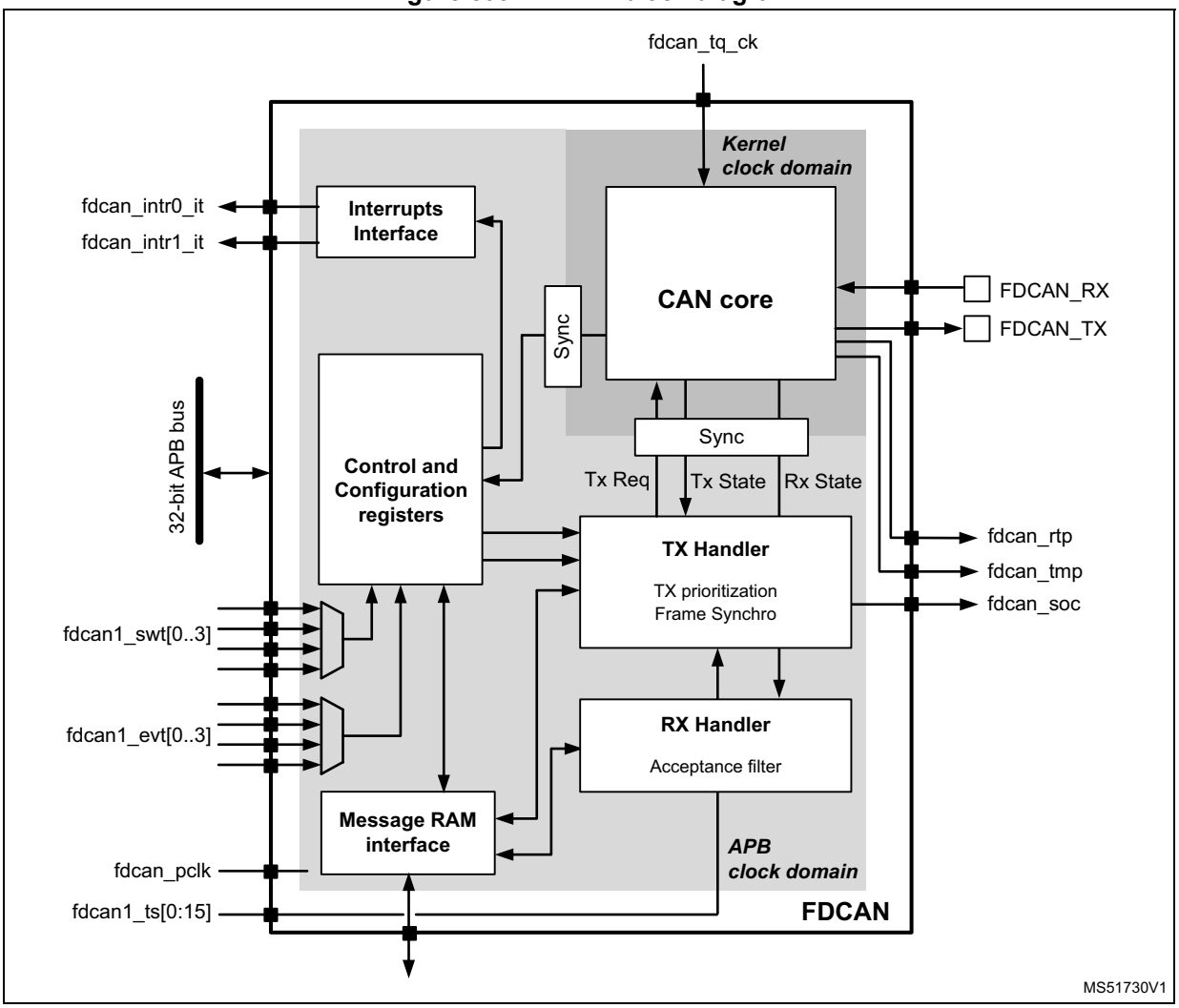
72.3 FDCAN implementation

Table 738. Main features

Module	FDCAN1	FDCAN2
TTCAN	X	-

72.4 FDCAN functional description

Figure 899. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it. By programming EINT0 and EINT1 bits in the FDCAN_ILE register, the interrupt lines can be enabled or disabled separately.

CAN core

The CAN core contains the protocol controller and the receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions, and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain, and vice versa.

Tx handler

Controls the message transfer from the message RAM to the CAN core. A maximum of 32 Tx buffers can be configured for transmission. Tx buffers can be used as dedicated Tx buffers, as Tx FIFO, part of a Tx queue, or as a combination of them. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

On FDCAN1, the Tx handler also implements the frame synchronization entity (FSE) which controls time-triggered communication according to ISO11898-4. It synchronizes itself with the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the message RAM. Stop watch trigger, event trigger, and time mark interrupt are synchronization interfaces.

Rx handler

Controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, each of configurable size, and up to 64 dedicated Rx buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx buffer, in contrast to a receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

APB interface

Connects the FDCAN to the APB bus.

Message RAM interface

Connects the FDCAN access to an external 10 Kbytes message RAM through a RAM controller/arbiter.

72.4.1 Operating modes

Software initialization

Software initialization is started by setting the INIT bit in the FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While the INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The counters of the error management logic (EML) are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards, the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in the FDCAN_CCCR register and the CCE bit in the FDCAN_CCCR register are set.

The CCE bit in the FDCAN_CCCR register can be set/cleared only while the INIT bit in the FDCAN_CCCR is set. The CCE bit in FDCAN_CCCR register is automatically cleared when the INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when the CCE bit in the FDCAN_CCCR register is set:

- FDCAN_HPMS - high priority message status
- FDCAN_ RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status
- FDCAN_TTOST - TT (time trigger) operation status (FDCAN1 only)
- FDCAN_TTLGT - TT local and global time, only global time FDCAN_TTLGT.GT is reset (FDCAN1 only)
- FDCAN_TTCTC - TT cycle time and count (FDCAN1 only)
- FDCAN_TTCSM - TT cycle sync mark (FDCAN1 only)

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit in FDCAN_CCCR is set.

The following registers can be written only when the CCE bit in the FDCAN_CCCR register is cleared:

- FDCAN_TXBAR - Tx buffer add request
- FDCAN_TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can be set only by software while both INIT bit and CCE bit in FDCAN_CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can be set/cleared only while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN1 default operating mode after hardware reset is event-driven CAN communication without time triggers (FDCAN_TTOCF.OM = 00). It is required that both INIT bit and CCE bit in the FDCAN_CCCR register are set before the TT operation mode can be changed.

Once the FDCAN is initialized and the INIT bit in the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the CAN FD protocol. The first is the long frame mode (LFM), where the data field of a CAN frame can be longer that eight bytes. The second variant is the fast frame mode (FFM), where the control, data, and CRC fields of a CAN frame are

transmitted with a higher bitrate than the beginning and the end of the frame. Fast frame mode can be used in combination with long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are now decoded as FDF bit. FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bitrate inside this CAN FD frame is switched. A CAN FD bitrate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for protocol expansions. If the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting bit FDCAN_PSR.PXE. When protocol exception handling is enabled (FDCAN_CCCR.PXHD = 0), this causes the operation state to change from receiver (FDCAN_PSR.ACT = 10) to integrating (FDCAN_PSR.ACT = 00) at the next sample point. In case protocol exception handling is disabled (FDCAN_CCCR.PXHD = 1), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming FDCAN_CCCR.FDOE. If FDCAN_CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of classic CAN frames is always possible. Whether a CAN FD or a classic CAN frame is transmitted, it can be configured via bit FDF in the respective Tx buffer element. With FDCAN_CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted, even if bit FDF of a Tx buffer element is set. FDCAN_CCCR.FDOE and FDCAN_CCCR.BRSE can only be changed while FDCAN_CCCR.INIT and FDCAN_CCCR.CCE are both set.

With FDCAN_CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 1, transmission of CAN FD frames with bitrate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bitrate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bitrate switching option for transmissions.
- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking must be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has been completed. Then all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. DLC codes 0 to 8 have the same coding as in standard CAN, codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 739*.

Table 739. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bitrate switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the bit timing and prescaler register FDCAN_NBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the fast bit timing and prescaler register FDCAN_DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bitrate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bitrate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long frames and CAN FD fast frames, the value of the bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for CAN FD transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives an FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx buffer or Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bitrate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bitrates during the CAN FD data phase, independent from the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit FDCAN_DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by FDCAN_TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of minimum time quanta (mtq, that is, one period of fdcan_tq_ck clock).

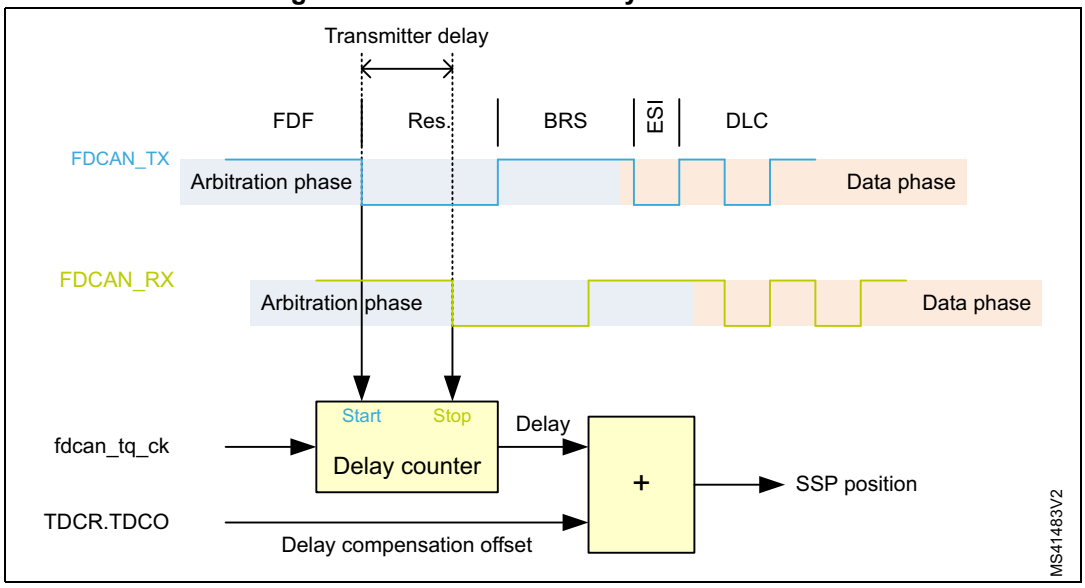
FDCAN_PSR.TDCV shows the actual transmitter delay compensation value, cleared when FDCAN_CCCR.INIT is set, and updated at each transmission of an FD frame while FDCAN_DBTP.TDC is set.

The following boundary conditions must be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than six bit times in the data phase.
- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than or equal to 127 mtq. If this sum exceeds 127 mtq, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs

If transmitter delay compensation is enabled by programming FDCAN_DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is 1 mtq.

Figure 900. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming FDCAN_TDCR.TDCF. This defines a minimum value for the SSP position.

Dominant edges on FDCANx_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least FDCAN_TDCR.TDCF and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, but waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (FDCAN_ECR.REC, FDCAN_ECR.TEC) are frozen while error logging (FDCAN_ECR.CEL) is active. The software can set the FDCAN into restricted operation mode by setting bit FDCAN_CCCR.ASM. The bit can be set only by software when both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave restricted operation mode, the software must reset FDCAN_CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bitrates. In this case the application tests different bitrates and leaves the restricted operation mode after it has received a valid frame.

FDCAN_CCCR.ASM is also controlled by the clock calibration unit. When the calibration process is enabled, the restricted operation mode is entered and the FDCAN_CCR.ASM bit is set. Once the calibration is completed, the FDCAN_CCR.ASM bit is cleared.

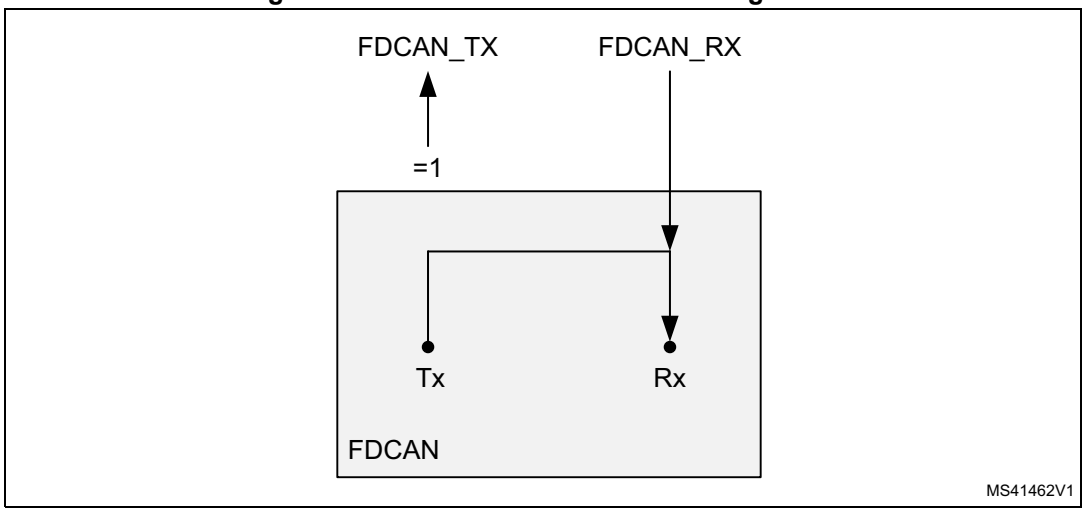
Note: The restricted operation mode must not be combined with the loop back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting FDCAN_CCCR.MON bit or when error level S3 (FDCAN_TTOST.EL = 11) is entered. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode register FDCAN_TXBRP is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 901* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 901. Pin control in bus monitoring mode



Disabled automatic retransmission (DAR) mode

According to the CAN Specification (see ISO11898-1, 6.3.3 recovery management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled.

To support time-triggered communication as described in ISO 11898-1: 2015, chapter 9.2, the automatic retransmission may be disabled via FDCAN_CCCR.DAR.

Frame transmission in disabled automatic retransmission (DAR) mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer Tx request pending bit FDCAN_TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set
- Arbitration loss or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx not set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via register FDCAN_CCCR.CSR. As long as the clock stop request is active, bit FDCAN_CCCR.CSR is read as 1.

When all pending transmission requests have been completed, the FDCAN waits until the bus idle state is detected. Then the FDCAN sets FDCAN_CCCR.INIT to 1 to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power down by setting FDCAN_CCCR.CSA to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to FDCAN_CCCR.INIT has no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application must turn on the module clocks before resetting the CC control register flag FDCAN_CCCR.CSR. The FDCAN acknowledges this by resetting FDCAN_CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit FDCAN_CCCR.INIT.

Test modes

To enable write access to FDCAN test register (see *Section 72.5.4*), bit FDCAN_CCCR.TEST must be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming FDCAN_TEST.TX. Additionally, to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from FDCAN_TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to FDCAN_TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via FDCAN_TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop back mode

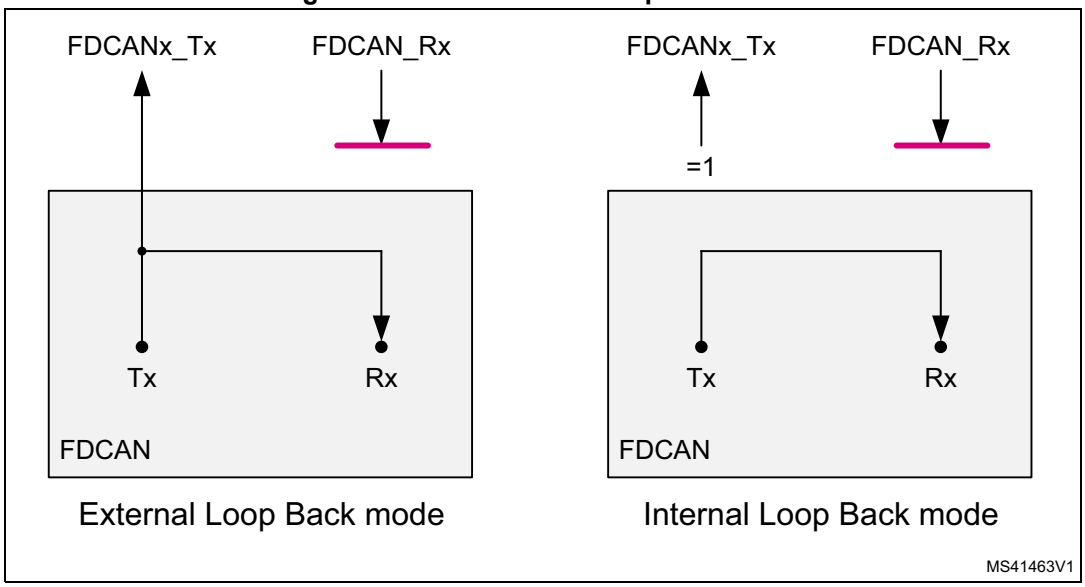
The FDCAN can be set in external loop back mode by programming FDCAN_TEST.LBCK to 1. In loop back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 902* (left side) shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits FDCAN_TEST.LBCK and FDCAN_CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, the FDCAN_RX pin is disconnected from the FDCAN and the FDCAN_TX pin is held recessive. *Figure 902* (right side) shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop back mode.

Figure 902. Pin control in loop back mode



Application watchdog (FDCAN1 only)

The application watchdog is served by reading register FDCAN_TTOST. When the application watchdog is not served in time, bit FDCAN_TTOST.AWE is set, all TTCAN communication is stopped, and the FDCAN1 is set into bus monitoring mode.

The TT application watchdog can be disabled by programming the application watchdog limit FDCAN_TTOCF.AWL to 0x00. The TT application watchdog must not be disabled in a TTCAN application program.

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wraparound counter. A prescaler FDCAN_TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 … 16). The counter is readable via FDCAN_TSCV.TCV. A write access to register FDCAN_TSCV resets the counter to 0. When the timestamp counter wraps around the interrupt flag FDCAN_IR.TSW is set.

On the start of frame reception/transmission the counter value is captured and stored into the timestamp section of an Rx buffer/Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit FDCAN_TSCC.TSS, a 16-bit timestamp can be used.

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by FDCAN_TSCC.TCP as the timestamp counter. The timeout counter is configured via register FDCAN_TOCC. The actual counter value can be read from FDCAN_TOCV.TOC. The timeout counter can only be started while FDCAN_CCCR.INIT = 0. It is stopped when FDCAN_CCCR.INIT = 1, for example when the FDCAN enters Bus_Off state.

The operation mode is selected by FDCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when FDCAN_CCCR.INIT is reset. A write to FDCAN_TOCV

presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the interrupt flag FDCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at FDCAN_TOCC.TOP.

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented can vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baudrate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

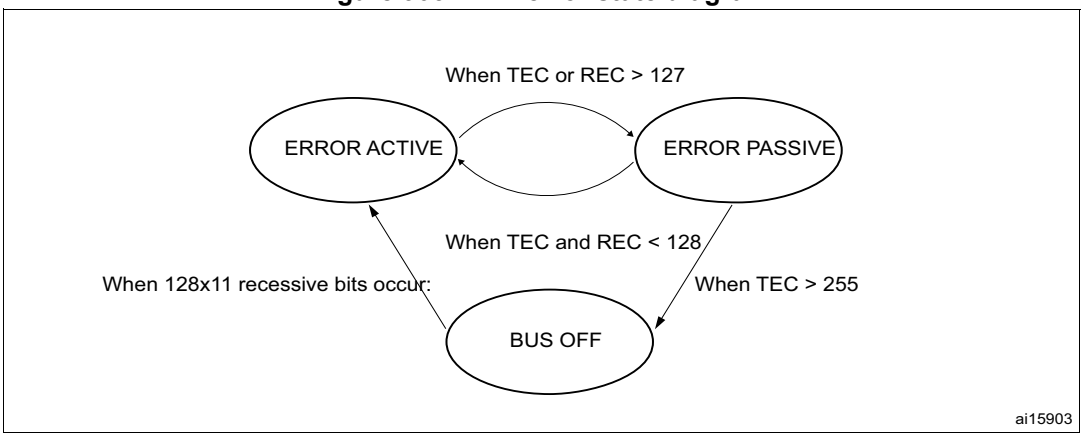
72.4.2 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a transmit error counter (TEC value, in the CAN_ECR register) and a receive error counter (REC value, in the CAN_ECR register), incremented or decremented according to the error condition. Both of them may be read by software to determine the stability of the network. For detailed information about TEC and REC management, refer to the CAN standard.

Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by the BO bit in the CAN_PSR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages. The CAN must wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

Figure 903. CAN error state diagram

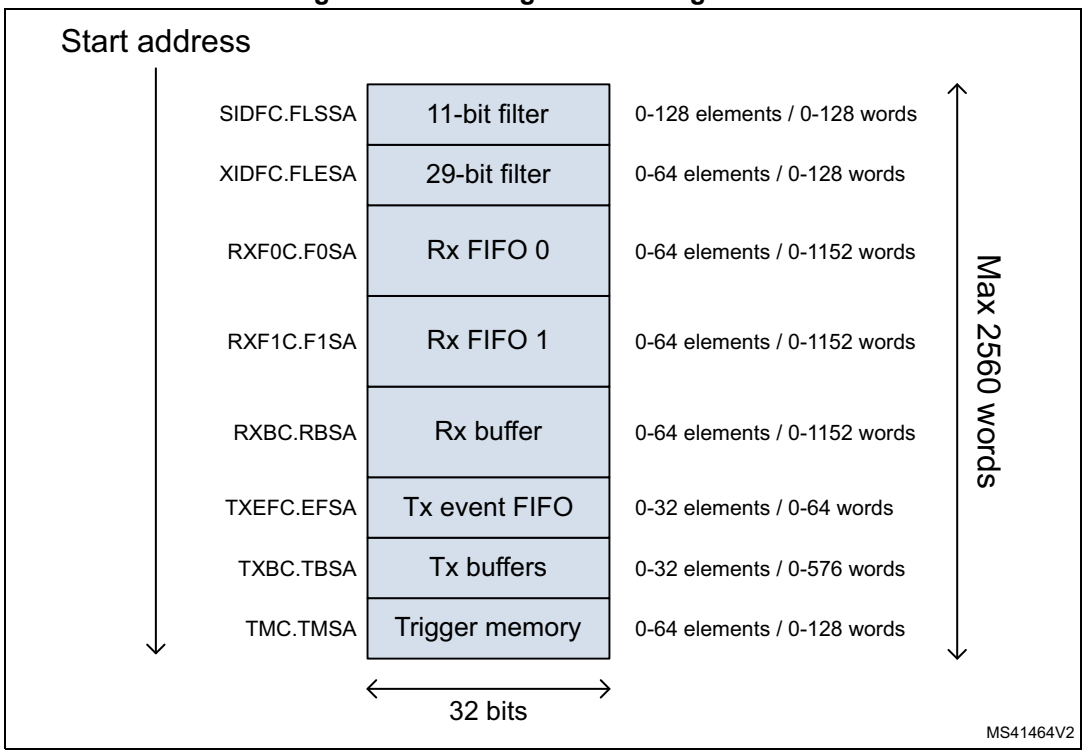


Note: In initialization mode, CAN does not monitor the CAN RX signal, therefore it cannot complete the recovery sequence. To recover, CAN must be in normal mode.

72.4.3 Message RAM

The message RAM has a width of 32 bits. The FDCAN module can be configured to allocate up to 2560 words in the message RAM. It is not necessary to configure each of the sections listed in *Figure 904*, nor is there any restriction with respect to the sequence of the sections.

Figure 904. Message RAM configuration



When the FDCAN addresses the message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The FDCAN does not check for erroneous configuration of the message RAM. In particular, the configuration of the start addresses of the different sections and the number of elements of each section must be done carefully to avoid falsification or loss of data.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx buffers or to 1 of the two Rx FIFOs, as well as the Rx FIFO put and get Indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from 0 to 128 elements for the 11-bit filter and from 0 to 64 for the 29-bit filter)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (FDCAN_GFC)
- Standard ID filter configuration (FDCAN_SIDFC)
- Extended ID filter configuration (FDCAN_XIDFC)
- Extended ID AND Mask (FDCAN_XIDAM)

Depending on the configuration of the filter element (SFEC / EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx buffer
- Store received frame in Rx buffer and generate pulse at filter event pin
- Reject received frame
- Set high priority message interrupt flag FDCAN_IR.HPM
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (for example CRC error), this message is discarded with the following impact:

- **Rx buffer**
  New data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC.
- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type, see FDCAN_PSR.LEC and FDCAN_PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* must be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independently of the filter(s) used. The result of the acceptance filter process depends strongly upon the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID / SF2ID and EF1ID / EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = 00: The message ID of received frames is AND-ed with the extended ID AND Mask (FDCAN_XIDAM) before the range filter is applied
- EFT = 11: The extended ID AND Mask (FDCAN_XIDAM) is not used for range filtering

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element must be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID / EF1ID is used as message ID filter, while SF2ID / EF2ID is used as filter mask.

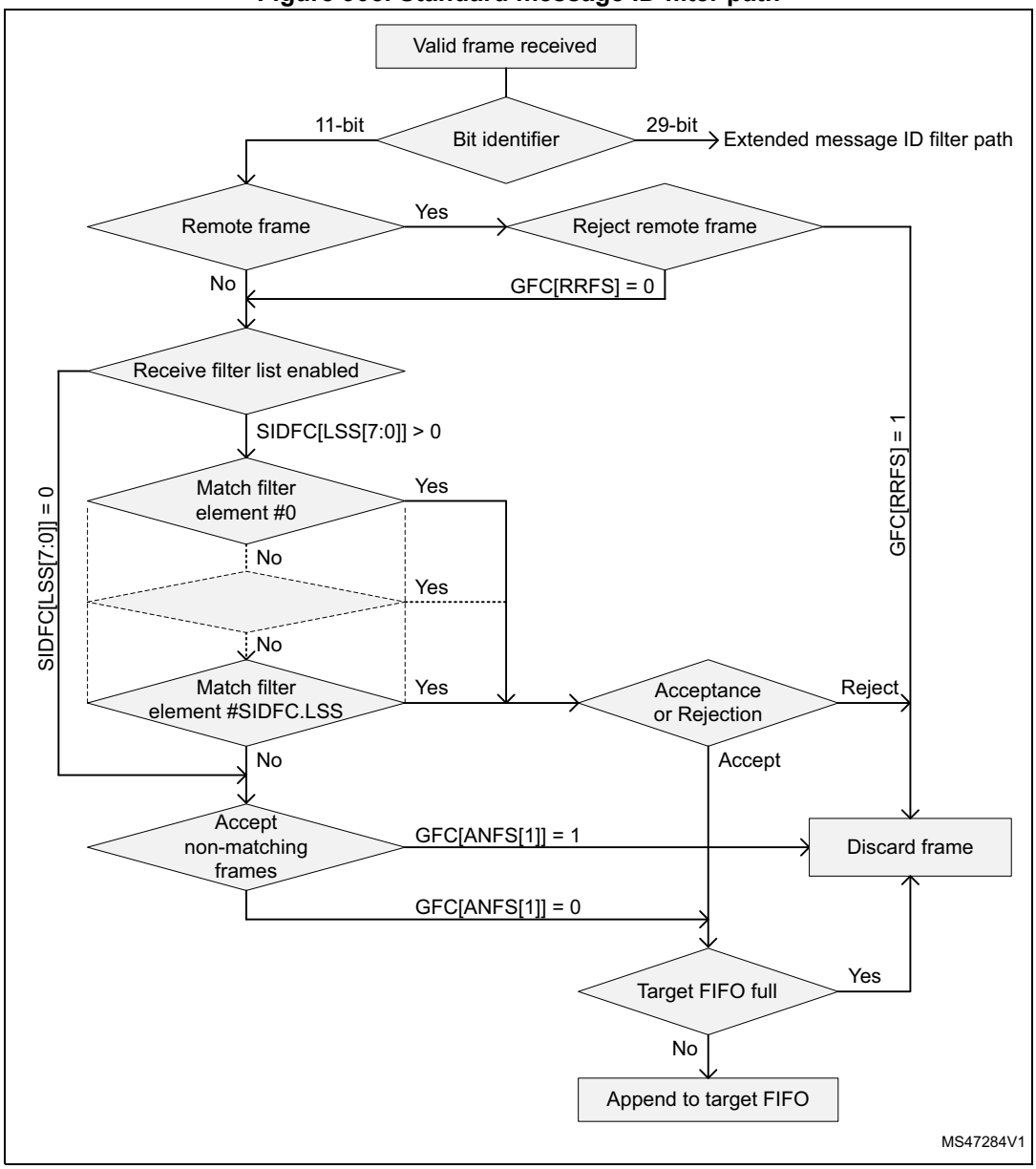
A 0 bit at the filter mask masks out the corresponding bit position of the configured ID filter, for example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering.

If all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 905 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 72.4.22*.

Figure 905. Standard message ID filter path

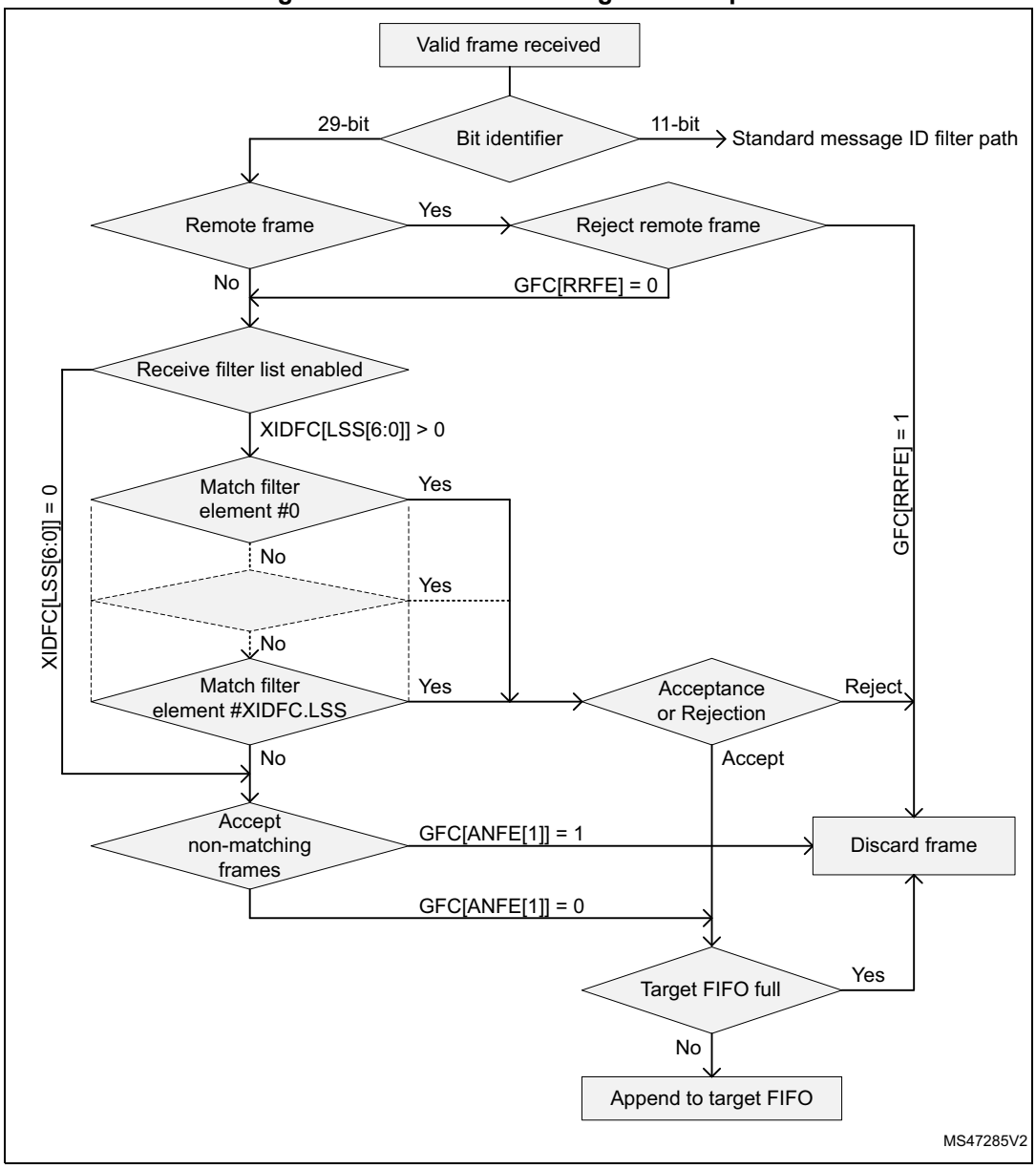


Controlled by the global filter configuration (FDCAN_GFC) and the standard ID filter configuration (FDCAN_SIDFC) message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Extended message ID filtering

Figure 906 shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 72.4.23*.

Figure 906. Extended message ID filter path



Controlled by the global filter configuration FDCAN_GFC and the extended ID filter configuration FDCAN_XIDFC message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The extended ID AND Mask (FDCAN_XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers FDCAN_RXF0C and FDCAN_RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx buffer and FIFO element are described in *Section 72.4.19*.

When an Rx FIFO full condition is signaled by FDCAN_IR.RFnF, no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the interrupt flag FDCAN_IR.RFnL is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FDCAN_RXFnC.FnWM, interrupt flag FDCAN_IR.RFnW is set.

When reading from an Rx FIFO, Rx FIFO get index RXFnS[FnGI] + FIFO element size must be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition, the interrupt flag FDCAN_IR.RFnF is set.

If a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition, the interrupt flag FDCAN_IR.RFnL is set.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is that it can happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 908* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case, the two messages stored in elements 1 and 2 are lost.

After reading from the Rx FIFO, the number of the last element read must be written to the Rx FIFO acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

Dedicated Rx buffers

The FDCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via FDCAN_RXBC.RBSA.

For each Rx buffer a standard or extended message ID filter element with SFEC / EFEC=111 and SFID2 / EFID2[10:9] = 00 must be configured (see *Section 72.4.22* and *Section 72.4.23*).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition, the flag FDCAN_IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

Table 740. Example of filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags must be reset by the user by writing a 1 to the respective bit position.

While an Rx buffer New data flag is set, a message ID filter element referencing this specific Rx buffer is not matched, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx buffer handling

- Reset interrupt flag FDCAN_IR.DRX
- Read New data registers
- Read messages from message RAM
- Reset New data flags of processed messages

Filtering for Debug messages

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC must be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New data flag nor FDCAN_IR.DRX are set. The reception of debug messages can be monitored via FDCAN_RXF1S.DMS.

Table 741. Example of filter configuration for Debug messages

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO.Up to 32 Tx buffers can be set up for message transmission (see *Dedicated Tx buffers*). Depending on the configuration of the element size (FDCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message data field.

Table 742. Possible configurations for frame transmission

FDCAN_CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bitrate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bitrate switching
1	1	1	1	FD with bitrate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register FDCAN_TXBRP is updated, or when a transmission has been started.

Transmit pause

This feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, after the first successful message transmission, it waits for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a

message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit in the FDCAN_CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (FDCAN_CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmission under complete control of the CPU. Each dedicated Tx buffer is configured with a specific message ID. In case that multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an add request via FDCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding the element size defined in *Table 743* times the transmit buffer index (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Table 743. Tx buffer/FIFO - Queue element size

FDCAN_TXESC.TBDS[2:0]	Data field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Tx FIFO

Tx FIFO operation is configured by programming FDCAN_TXBC.TFQM to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index FDCAN_TXFQS.TFGI. After each transmission the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level FDCAN_TXFQS.TFFL as the difference between get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages must be written to the Tx FIFO starting with the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (FDCAN_TXFQS.TFQF = 1) is signaled. In this case, no further messages should be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via FDCAN_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO free level remain unchanged.

A TX FIFO element allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx FIFO buffer is calculated by adding element size times the put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Tx queue

Tx queue operation is configured by programming FDCAN_TXBC.TFQM to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). If multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request cyclically increments the put index to the next free Tx buffer. In case that the Tx queue is full (FDCAN_TXFQS.TFQF = 1), the put index is not valid and no further message should be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

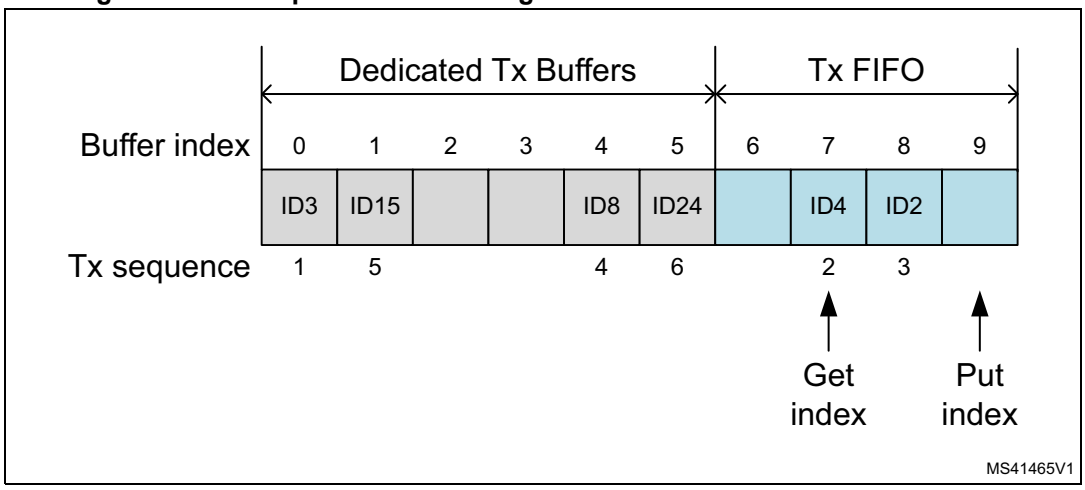
The application may use register FDCAN_TXBRP instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding element size times the Tx queue put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Mixed dedicated Tx buffers / Tx FIFO

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx buffers assigned to the Tx FIFO is configured by FDCAN_TXBC.TFQS. In case, FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 907. Example of mixed configuration dedicated Tx buffers / Tx FIFO



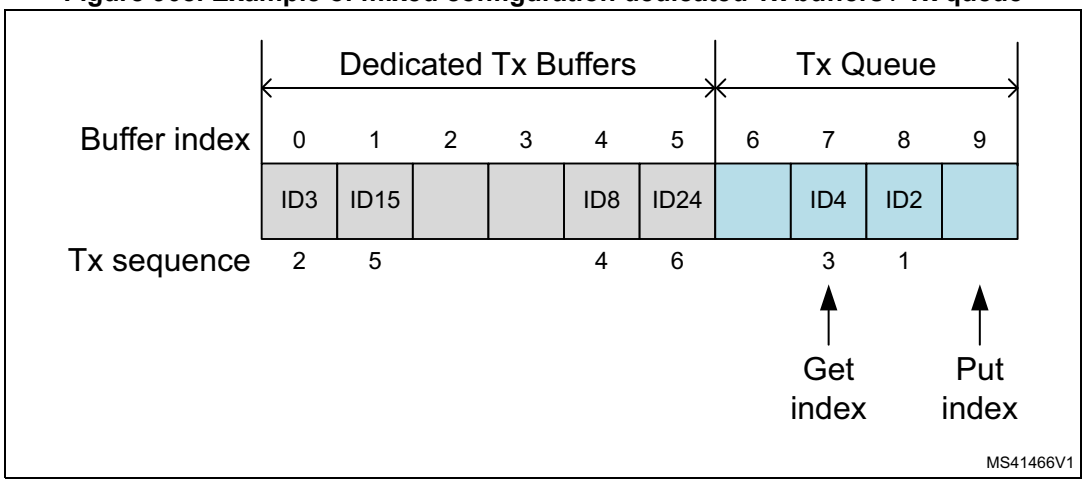
Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by FDCAN_TXFS.TFGI)
- Buffer with lowest message ID gets highest priority and is transmitted next

Mixed dedicated Tx buffers / Tx queue

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx queue buffers is configured by FDCAN_TXBC.TFQS. If FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 908. Example of mixed configuration dedicated Tx buffers / Tx queue



Tx priority setting:

- Scan all Tx buffers with activated transmission request
- Tx buffer with lowest message ID gets highest priority and is transmitted next

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the user must write a 1 to the corresponding bit position (= number of Tx buffer) of register FDCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register FDCAN_TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: If a pending transmission is canceled immediately before this transmission starts, a short time window follows where no transmission is started even if another message is pending in this node. This may enable another node to transmit a message that may have a priority lower than that of the second message in this node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, the message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in *Tx FIFO*. Depending on the configuration of the element size (FDCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message data field.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, i.e. a Tx buffer holds only the message to transmit, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by FDCAN_IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the interrupt flag FDCAN_IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by FDCAN_TXEFC.EFWM, interrupt flag FDCAN_IR.TEFW is set.

When reading from the Tx event FIFO, two times the Tx event FIFO get index FDCAN_TXEFS.EFGI must be added to the Tx event FIFO start address FDCAN_TXEFC.EFSA.

72.4.4 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index, see *FDCAN Rx FIFO 0 acknowledge register*

(FDCAN_RXF0A), *FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)*, and *FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)*. Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care must be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFO acknowledge index should not be written because this would set the get index to a wrong position and also alters the FIFO fill level. In this case, some of the older FIFO elements would be lost.

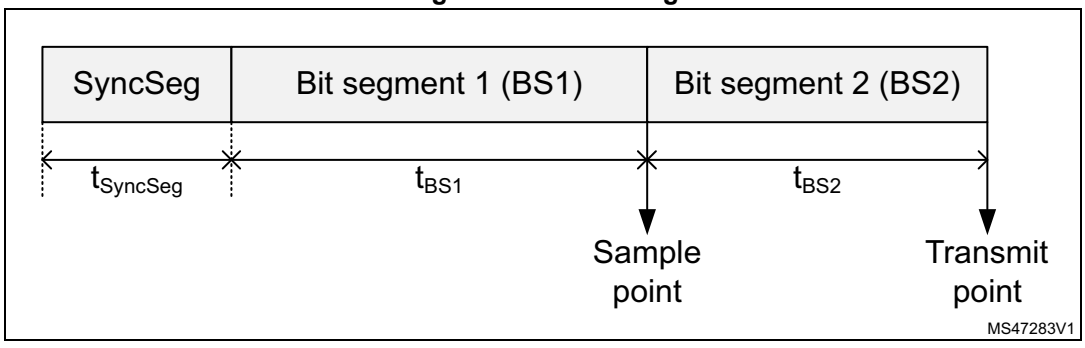
Note: The application must ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

72.4.5 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges. As shown in *Figure 909*, its operation may be explained simply by splitting the bit time in three segments:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, which has a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.

Figure 909. Bit timing



The baudrate is the inverse of bit time (baudrate = 1 / bit time), which, in turn, is the sum of three components. *Figure 909* indicates that bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$, where:

- for the nominal bit time
  - tq = (FDCAN_NBTP.NBRP[8:0] + 1) * $t_{fdcan_tq_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_NBTP.NTSEG1[7:0] + 1)
  - t_{BS2} = tq * (FDCAN_NBTP.NTSEG2[6:0] + 1)
- for the data bit time
  - tq = (FDCAN_DBTP.DBRP[4:0] + 1) * $t_{fdcan_tq_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_DBTP.DTSEG1[4:0] + 1)
  - t_{BS2} = tq * (FDCAN_DBTP.DTSEG2[3:0] + 1)

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. Registers FDCAN_DBTP and FDCAN_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set.

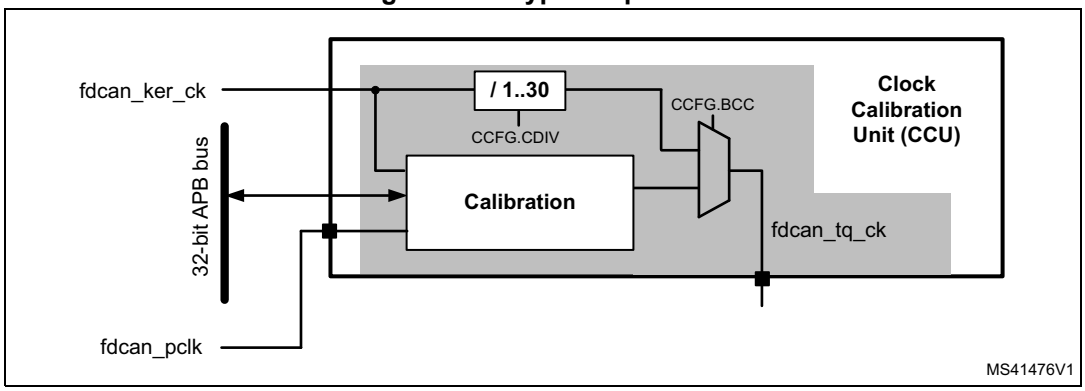
Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

72.4.6 Clock calibration on CAN

After device reset the clock calibration unit (CCU) does not provide a valid clock signal to the FDCAN(s). The CCU must be initialized via FDCAN_CCFG register. The FDCAN_CCFG register can be written only when FDCAN1 has both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set. In consequence the CCU and the FDCAN1 initialization needs to be completed before any FDCAN module can operate.

Clock calibration is bypassed when FDCAN_CCFG.BCC = 1 (see *Figure 910*).

Figure 910. Bypass operation



Operating conditions

The clock calibration on CAN unit is designed to operate under the following conditions:

- a CAN kernel clock frequency fdcan_ker_ck up of at least 80 MHz
- FDCAN bitrates:
  - Nominal bitrate: up to 1 Mbit/s
  - Data bitrate: between 125 kbps and 1Mbps

The clock calibration on FDCAN unit generates a calibrated time quanta clock fdcan_tq_ck in the range from 0.5 to 25 MHz.

Note: The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock (fdcan_tq_ck < fdcan_pclk). This must be considered when the clock calibration on CAN unit is bypassed (FDCAN_CCFG.BCC = 1).

Calibration accuracy

The calibration accuracy in state Precision_Calibrated depends upon the factors listed below.

- Dynamic clock tolerance at the CAN kernel clock input fdcan_ker_ck
- Measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one fdcan_pclk period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of FDCAN_CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages must be chosen to fit the clock tolerance requirements of the FDCAN1 module.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages for example caused by change of temperature or operating voltage.

Functional description

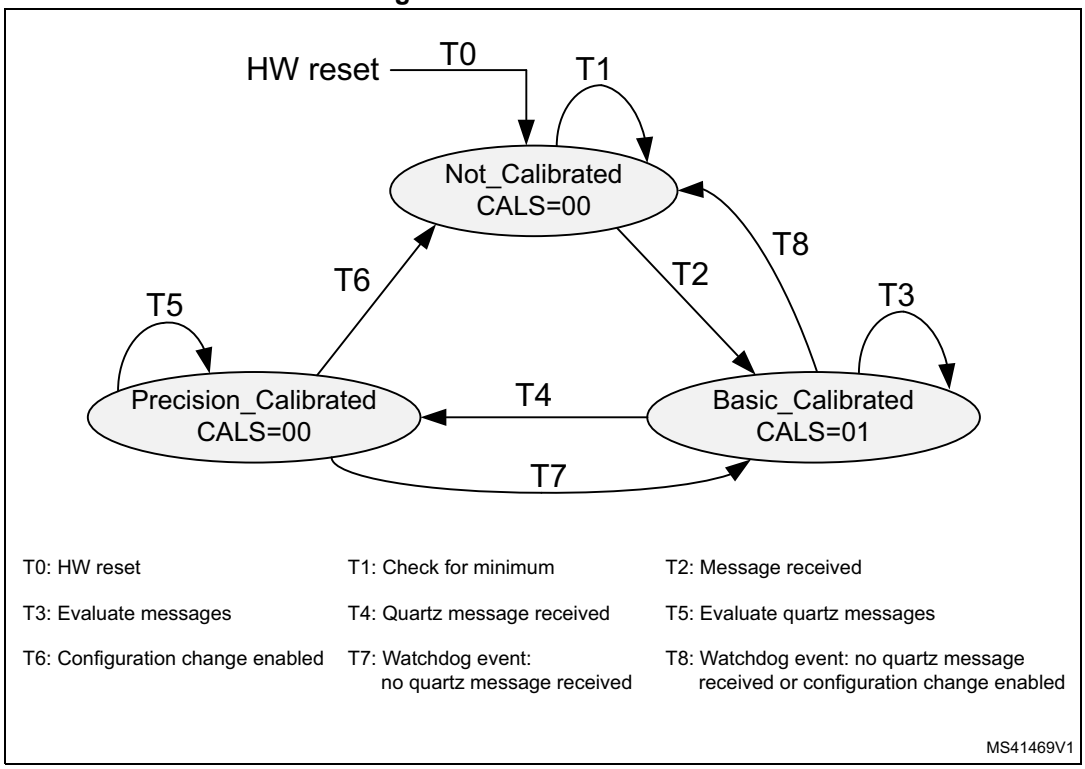
Calibration of the time quanta clock fdcan_tq_ck via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock fdcan_ker_ck.

1. First step: basic calibration
   The minimum distance between two consecutive falling edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider (FDCAN_CCFG.CDIV) is updated each time a new

measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message.

2. Second step: Precision calibration
The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of fdcan_ker_ck periods. The length of this bit sequence can be configured to 32 or 64 bits via FDCAN_CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6-byte data field is required. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 911. FSM calibration



A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE. it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

Until precision calibration is achieved, FDCANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the fdcan_ker_clk is done by software by evaluating the calibration status from register FDCAN_CCU_CSTAT, FDCANs must be set to restricted operation mode (FDCAN_CCCR.ASM = 1) until the calibration on CAN unit is in state Precision_Calibrated (see *Application*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. calibration frames are detected by the FDCAN1 acceptance filtering A filter element and a Rx buffer must be configured in the FDCAN1 to identify and store calibration messages. After reception of a calibration message the Rx buffer new data flag must be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node must transmit its first message after startup with at least one 1010 binary sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the clock calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the FDCAN1 complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

Configuration

The clock calibration on CAN unit is configured via register FDCAN_CCFG, that is, when FDCAN1 has FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin FDCAN1_RX is measured. The number of clock periods depends on the clock frequency applied at input fdcan_ker_ck. In case the measured number of clock periods is below the minimum configured by FDCAN_CCFG.OCPM (as an example, because of a glitch on FDCAN1_RX) the value is discarded and measurement continues.

It is recommended to configure FDCAN_CCFG.OCPM slightly below two CAN bit times:

FDCAN_CCFG.OCPM < ((2 x CAN bit time) / fdcan_ker_ck period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via FDCAN_CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages.

The number of time quanta per bit time configured by FDCAN_CCFG.TQBT is used together with the measured number of oscillator clock periods FDCAN_CCU_CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring FDCAN_CCFG.BCC = 1, the internal clock divider must be configured via FDCAN_CCFG.CDIV to fulfill the condition fdcan_tq_ck < fdcan_pclk.

Note: When clock calibration on CAN is active (FDCAN_CCFG.BCC = 0), the baudrate prescalers of FDCAN modules must be configured to inactive.

Status signaling

The status of the clock calibration on CAN unit can be monitored by reading register FDCAN_CCU_CSTAT. When in state Precision_Calibrated the oscillator clock period counter FDCAN_CCU_CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while FDCAN_CCU_CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by FDCAN_CCU_CSTAT.CALS. A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC. If enabled by the interrupt enable FDCAN_CCU_IE.CSCE it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE (set to high), it remains active until reset by FDCAN_CCU_IE.CWE.

72.4.7 Application

Clock calibration bypassed

The CCU internal clock divider is configured for division by one (FDCAN_CCFG.CDIV = 0x0000). In this operation mode the input clock fdcan_ker_ck is directly routed to the clock output fdcan_tq_ck. In this case fdcan_tq_ck is independent from the configuration and status of FDCAN1and FDCAN2 connected to the CCU. CAN FD operation is possible with an FDCAN_ker_ck above 80 MHz.

Software calibration

The clock calibration on CAN unit also supports software calibration of fdcan_ker_ck by trimming of an on-chip oscillator. For calculation of the trimming values the user must read the CCU state from FDCAN_CCU_CSTAT. The clock from fdcan_ker_ck is routed to output fdcan_tq_ck (FDCAN_CCFG.BCC = 1).

The input clock fdcan_ker_ck must be at least 80 MHz. The clock divider of CCU must be configured via FDCAN_CCFG.CDIV to bring fdcan_tq_ck to a valid range. All other configuration parameters must be set via FDCAN_CCFG. For correct operation of FDCAN1 and FDCAN2, the APB clock fdcan_pclk needs to be equal to or higher than the time quanta clock (fdcan_tq_ck). CAN FD operation is not possible.

For startup FDCAN modules must be both configured for restricted operation (FDCAN_CCCR.ASM = 1) before FDCAN_CCCR.INIT is reset. The input clock fdcan_ker_ck must be adjusted until the clock calibration on CAN unit has reached state Precision_Calibrated. Now the software can reset FDCAN_CCCR.ASM, and the CANFD1 and CANFD2 can start normal operation.

During operation the software must check regularly whether the clock calibration on CAN unit is still in state Precision_Calibrated. In case the clock calibration on CAN unit has left state Precision_Calibrated due to drift of fdcan_ker_ck, FDCAN modules must be set into restricted operation mode by programming FDCAN_CCCR.INIT, FDCAN_CCCR.CCE, and FDCAN_CCCR.ASM to 1. After fdcan_ker_ck has been adjusted successfully (clock calibration on CAN unit is in state Precision_Calibrated), FDCAN modules can resume normal operation.

Note: Trimming accuracy must be to sufficient to meet the CAN clock tolerance requirements for the configured bitrate.

Clock calibration active

This operation mode is entered by resetting FDCAN_CCFG.BCC to 0. In this operation mode the fdcan_ker_ck is controlled by the CCU.

The generation of CCU output signal fdcan_tq_ck depends upon the state of the FDCAN1. Input clock fdcan_ker_ck must be above 80 MHz. Configuration of the CCU and FDCAN1 is required. CAN FD operation is not possible.

If FDCAN1 turns to Bus_Off or when its INIT bit is set by the user command (FDCAN_CCCR.INIT = 1), the CCU enters state Not_Calibrated. CANFD1 and CANFD2 enter restricted operation mode.

Note: This is the default operation mode after reset in case the reset value of FDCAN_CCFG.BCC is configured to 0.

72.4.8 TTCAN operations (FDCAN1 only)

Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the time master (sender of the reference message).

For level 1 the data length must be at least one; for levels 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register FDCAN_TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Level 1

Level 1 operation is configured via FDCAN_TTOCF.OM = 01 and FDCAN_TTOCF.GEN. external clock synchronization is not available in level 1. The information related to the reference message is stored in the first data byte as shown in *Table 744*. Cycle_Count is optional.

Table 744. First byte of level 1 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					

Level 2

Level 2 operation is configured via FDCAN_TTOCF.OM = 10 and FDCAN_TTOCF.GEN. The information related to the reference message is stored in the first four data bytes as shown in *Table 745*. Cycle_Count and the lower four bits of FDCAN_NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 745. First four bytes of level 2 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

Level 0

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. External event-synchronized time-triggered operation is not available in level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In level 0 Next_is_Gap is always 0. Cycle_Count and the lower four bits of NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 746. First four bytes of level 0 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

72.4.9 TTCAN configuration

TTCAN timing

The network time unit (NTU) is the unit in which all times are measured. The NTU is a constant of the whole network and is defined as a priority by the network system designer. In TTCAN level 1 the NTU is the nominal CAN bit time. In TTCAN level 0 and level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN level 0 and level 2 the length of the NTU is defined by the time unit Ratio TUR. The TUR is in general a non-integer number, given by
TUR = FDCAN_TURNA.NAV / FDCAN_TURCF.DC. The NTU length is given by
NTU = CAN clock period x TUR.

The TUR numerator configuration NC is an 18-bit number, FDCAN_TURCF[NCL[15:0]] can be programmed in the range 0x0000 to 0xFFFF. FDCAN_TURCF[NCH[17:16]] is hard wired to 0b01. When 0xnnnn is written to FDCAN_TURCF[NCL[15:0]], FDCAN_TURNA.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR denominator configuration FDCAN_TURCF.DC is a 14-bit number. FDCAN_TURCF.DC may be programmed in the range 0x0001 to 0x3FFF (0x0000 is an illegal value).

In level 1, NC must be equal to or higher than 4 x FDCAN_TURCF.DC. In level 0 and level 2 NC must be equal to or higher than 8 x FDCAN_TURCF.DC to get the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets FDCAN_TURCF.DC to 0x1000 and FDCAN_TURCF.NCL to 0x10000, resulting in an NTU consisting of sixteen CAN clock periods. Local time and application watchdog are not started before either the FDCAN_CCCR.INIT is reset, or FDCAN_TURCF.ELT is set. FDCAN_TURCF.ELT may not be set before the NTU is configured. Setting FDCAN_TURCF.ELT to 1 also locks the write access to register FDCAN_TURCF.

At startup FDCAN_TURNA.NAV is updated from NC (= FDCAN_TURCF.NCL + 0x10000) when FDCAN_TURCF.ELT is set. In TTCAN level 1 there is no drift compensation. FDCAN_TURNA.NAV does not change during operation, it is always equal to NC.

In TTCAN level 0 and level 2 there are two possibilities for FDCAN_TURNA.NAV to change. When operating as time slave or backup time master, and when FDCAN_TTOCF.ECC is set, FDCAN_TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization, it returns to NC. When operating as the actual time master, and when FDCAN_TTOCF.EECS is set, the user may update FDCAN_TURCF.NCL. When the user sets FDCAN_TTOCN.ECS, FDCAN_TURNA.NAV is updated from the new value of NC at the next reference message. The status flag FDCAN_TTOST.WECS as is set when FDCAN_TTOCN.ECS is set and is cleared when FDCAN_TURNA.NAV is updated. FDCAN_TURCF.NCL is write locked while FDCAN_TTOST.WECS is set.

In TTCAN level 0 and level 2 the clock calibration process adapts FDCAN_TURNA.NAV in the range of the synchronization deviation limit SDL of NC ± 2(FDCAN_TTOCF.LDSDL + 5). FDCAN_TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of FDCAN_TURNA.NAV.

The synchronization deviation SD is the difference between NC and FDCAN_TURNA.NAV (SD = | NC - FDCAN_TURNA.NAV |). It is limited by the synchronization deviation limit SDL, which is configured by its dual logarithm FDCAN_TTOCF.LDSDL (SDL = 2 (FDCAN_TTOCF.LDSDL + 5)) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new basic cycle. When the calculated TURNA[NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and FDCAN_TTIR.GTE is set and FDCAN_TTOSC.QCS is reset, or in case of the Disc_Bit = 1, FDCAN_TTIR.GTD is set.

Table 747. TUR configuration example

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
FDCAN _TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

FDCAN_TTOCN.ECS schedules NC for activation by the next reference message. FDCAN_TTOCN.SGT schedules FDCAN_TTGTP.TP for activation by the next reference message. Setting FDCAN_TTOCN.ECS and FDCAN_TTOCN.SGT requires FDCAN_TTOCF.EECS to be set (external clock synchronization enabled) while the FDCAN is actual time master.

The TTCAN module provides an application watchdog to verity the function of the application program. The user must serve this watchdog regularly, else all CAN bus activity is stopped. The application watchdog limit FDCAN_TTOCF.AWL specifies the number of NTUs between two times the watchdog must be served. The maximum number of NTUs is 256. The application watchdog is served by reading register FDCAN_TTOST. FDCAN_TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, the interrupt flag FDCAN_TTIR.AW is set. For software development, the application watchdog may be disabled by programming FDCAN_TTOCF.AWL to 0x00, see *Section 72.6.3*.

Timing of interface signals

The timing events that cause a pulse at output FDCAN trigger time mark interrupt pulse fdcan1_tmp for more than one instance and fdcan_tmp if only one instance; FDCAN register time mark interrupt pulse fdcan1_rtp for more than one instance and fdcan_rtp if only one instance are generated in the CAN clock domain.

There is a clock domain crossing delay to be considered before the same event is visible in the APB clock domain (when FDCAN_TTIR.TTMI is set or FDCAN_TTIR.RTMI is set). As an example, the signals can be connected to the timing input(s) of another FDCAN node (fdcan_swt/fdcan_evt), in order to automatically synchronize two TTCAN networks.

Output FDCAN start of cycle_fdcan1_soc for more than one instance and cycle_fdcan1_soc if only one instance gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the APB clock domain.

72.4.10 Message scheduling

FDCAN_TTOCF.TM controls whether the TTCAN operates as potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message identifier FDCAN_TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest. There cannot be two nodes in the network using the same master priority. FDCAN_TTRMC.RID is used for recognition of reference messages. FDCAN_TTRMC.RMPS is not relevant for time slaves.

The initial reference trigger offset FDCAN_TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for FDCAN_TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

FDCAN_TTOCF.OM decides whether the node operates in TTCAN level 0, level 1, or level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on level 1 in a level 2 network, but not vice versa. The configuration of the TTCAN operation mode via FDCAN_TTOCF.OM is the last step in the setup. With FDCAN_TTOCF.OM = 00 (event-driven CAN communication), the FDCAN operates

according to ISO 11898-1: 2015, without time triggers.With FDCAN_TTOCF.OM = 01 (level 1), the FDCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With FDCAN_TTOCF.OM = 10 (level 2), the TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With FDCAN_TTOCF.OM = 11 (level 0), the FDCAN operates as event-driven CAN but maintains a calibrated global time base as in level 2.

FDCAN_TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in levels 0 and level 2 only) the global clock phase to an external reference.

FDCAN_TTMLM.ENTT in the TT matrix limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger repeat factors must be taken into account.

An inaccurate configuration of FDCAN_TTMLM.ENTT results either in a TxCount Underflow (FDCAN_TTIR.TXU = 1 and FDCAN_TTOST.EL = 01, severity 1), or in a Tx count Overflow (FDCAN_TTIR.TXO = 1 and FDCAN_TTOST.EL = 10, severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count 0, this node may finish its first matrix cycle with its Tx count resulting in a Tx count underflow condition. As long as a node is in state Synchronizing, its Tx_Triggers do not lead to transmissions.

FDCAN_TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of eight basic cycles FDCAN_TTMLM.CCM would be 7. FDCAN_TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

FDCAN_TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window message, the transmission cannot be started in that time window at all. FDCAN_TTMLM.TXEW must be chosen with respect to the network synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

Trigger memory

The trigger memory is part of the external message RAM to which the TTCAN is connected to (see *Section 72.5.26*). It stores up to 64 trigger elements. A trigger memory element consists of time mark TM, cycle code CC, trigger type TYPE, filter type FTYPE, message number MNR, message status count MSC, time mark event internal TMIN, time mark event external TMEX (see *Section 72.4.24*).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory must be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a

transmit trigger, the Tx handler starts to read the message from the message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx handler must be able to prepare the transmission before the transmit trigger time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mar k is reached. If the reference message is n NTU long, a trigger with a time mark lower than n never becomes active and is treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger time mark. The FDCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value lower than n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger time mark + 1 NTU + 1 CAN bit time.

The trigger list is different for all nodes in the CAN FD network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger types

Tx_Ref_Trigger (TYPE = 0000) and Tx_Ref_Trigger_Gap (TYPE = 0001) cause the transmission of a reference message by a time master. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized
time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Tx_Trigger_Single (TYPE = 0010), Tx_Trigger_Continuous (TYPE = 0011), Tx_Trigger_Arbitration (TYPE = 0100), and Tx_Trigger_Merged (TYPE = 0101) cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit is reset.

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code must be considered when the expected number of Tx_Triggers (FDCAN_TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = 0110) and Watch_Trigger_Gap (TYPE = 0111) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Rx_Trigger (TYPE = 1000) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger must be placed after the end of that message transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger time mark is reached.

Time_Base_Trigger (TYPE = 1001) are used to generate internal/external events depending on the configuration of TMIN and TMEX.

End_of_List (TYPE = 1010 … 1111) is an illegal trigger type, a configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

Restrictions for the node trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) never become active. The watch triggers themselves do not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) must be set to trigger type End_of_List.

A typical trigger list for a potential time master begins with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave is the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of

a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the reference trigger offset FDCAN_TTOCF.IRTO. In case error level 2 is reached (FDCAN_TTOST.EL = 10), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- APB clock frequency
- Speed and waiting time for trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for trigger handling

The example shows how the trigger list is derived from a node system matrix. Assumption is that node A is first time master and has knowledge of the section of the system matrix shown in *Table 748*.

Table 748. System matrix, Node A

Cycle count	Time mark						
	1	2	3	4	5	6	7
0	Tx7	-	-	-	-	TxRef	Error
1	Rx3	-	Tx2, Tx4		-	TxRef	Error
2	-	-	-	-	-	TxRef	Error
3	Tx7	-	Rx5	-	-	TxRef	Error
4	Tx7	-	-	Rx6	-	TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the corresponding number of basic cycles in the system matrix is, respectively, 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by FDCAN_TTMLM.CCM. The cycle code CC is composed of repeat factor (= value of most significant 1) and the number of the first basic cycle in the system matrix (= bit field after most significant 1).

As an example, with a cycle code of 0b0010011 (repeat factor = 16, first basic cycle = 3) and a maximum cycle count of FDCAN_TTMLM.CCM = 0x3F matches occur at cycle counts 3, 19, 35 and 51.

A trigger element consists of time mark TM, cycle code CC, trigger type TYPE, and message number MNR. For transmission MNR references the Tx buffer number (0 ... 31). For reception MNR references the number of the filter element (0 ... 127) that matched during acceptance filtering. Depending on the configuration of the filter type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition, a trigger element can be configured for generation of time mark event internal TMIN, and time mark event external TMEX. The message status count MSC holds the counter value (0 ... 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 749. Trigger list, Node A

Trigger	Time mark TM[15:0]	Cycle code CC [6:0]	Trigger type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	N/A	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	N/A	Watch_Trigger	N/A
9	N/A	N/A	End_of_List	N/A

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from 0 until a trigger with time mark higher than cycle time and with its cycle code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the time mark TM, the action defined by trigger type TYPE and message number MNR is started. There is an error in the configuration when End_of_List is reached.

At mark 6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the watch trigger at mark 7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of configuration errors

A configuration error is signaled via FDCAN_TTOST.EL = 11 (severity 3) when:

- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a

Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

- The FSE of a node with FDCAN_TTOCF.TM=0 (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.
- Any time mark placed inside the Tx enable window (defined by FDCAN_TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.
- A time mark is placed near the time mark of a Tx_Ref_Trigger and the reference trigger offset FDCAN_TTOST.RTO causes a reversal of their sequential order measured in cycle time.

TTCAN schedule initialization

The synchronization to the TTCAN message schedule starts when FDCAN_CCCR.INIT is reset. The TTCAN can operate strictly time-triggered (FDCAN_TTOCF.GEN = 0) or external event-synchronized time-triggered (FDCAN_TTOCF.GEN = 1). All nodes start with cycle time 0 at the beginning of their trigger list with FDCAN_TTOST.SYS = 00 (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode ignore Tx_Ref_Trigger and Watch_Trigger and use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time slaves

After configuration, a time slave ignores its Watch_Trigger and Watch_Trigger_Gap if it does not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to acknowledge or to send error flags). The first received reference message restarts the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter that counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In the bus monitoring mode it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the user must set FDCAN_CCCR.INIT = 1. After reset of FDCAN_CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets FDCAN_TTOST.SYS = 01 (Synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential time masters

After configuration, a potential time master starts the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It ignores its Watch_Trigger and Watch_Trigger_Gap when it does not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches

Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting FDCAN_TTIR.IWT re-enables the transmission of reference messages until the next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE is not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), sets interrupt flag FDCAN_TTIR.WT, switches off its CAN bus output, and enters the bus monitoring mode (FDCAN_CCCR.MON set to 1). In bus monitoring mode, it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets FDCAN_TTOST.SYS = 01 (synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (FDCAN_TTOST.MS = 11) when it was the transmitter of the last reference message, else it is backup time master (FDCAN_TTOST.MS = 10).

When all potential time masters have finished configuration, the node with the highest time master priority in the network becomes the current time master.

72.4.11 TTCAN gap control

All functions related to gap control apply only when the FDCAN is operated in external event synchronized time-triggered mode (FDCAN_TTOCF.GEN = 1). In this operation mode the FDCAN message schedule may be interrupted by inserting gaps between the basic cycles of the system matrix. All nodes connected to the CAN network must be configured for external event- synchronized time-triggered operation.

During a gap, all transmissions are stopped and the CAN bus remains idle. A gap is finished when the next reference message starts a new basic cycle. A gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = 1 for example gaps are initiated by the current time master.

The current time master has two options to initiate a gap. A gap can be initiated under software control when the application program writes FDCAN_TTOCN.NIG = 1. The Next_is_Gap bit is transmitted as 1 with the next reference message. A gap can also be initiated under hardware control when the application program enables the event trigger input pin fdcan_evt by writing FDCAN_TTOCN.GCS = 1. When a reference message is started and FDCAN_TTOCN.GCS is set, a HIGH level at event trigger pin fdcan_evt sets Next_is_Gap = 1.

As soon as that reference message is completed, the FDCAN_TTOST.WFE bit announces the gap to the time master as well as to the time slaves. The current basic cycle continues until its last time window. The time after the last time window is the gap time.

For the actual time master and the potential time masters, FDCAN_TTOST.GSI is set when the last basic cycle has finished and the gap time starts. In nodes that are time slaves, bit FDCAN_TTOST.GSI remains at 0.

When a potential time master is in synchronization state In_Gap (FDCAN_TTOST.SYS = 10), it has four options to intentionally finish a gap:

1. Under software control by writing FDCAN_TTOCN.FGP = 1.
2. Under hardware control (FDCAN_TTOCN.GCS = 1) an edge from HIGH to LOW at the event-trigger input pin fdcan_evt sets FDCAN_TTOCN.FGP and restarts the schedule.
3. The third option is a time-triggered restart. When FDCAN_TTOCN.TMG = 1, the next register time mark interrupt (FDCAN_TTIR.RTMI = 1) sets FDCAN_TTOCN.FGP and starts the reference message.
4. Finally any potential time master finishes a gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

None of these options can cause a basic cycle to be interrupted with a reference message.

Setting of FDCAN_TTOCN.FGP after the gap time begins starts the transmission of a reference message immediately and thereby synchronizes the message schedule. When FDCAN_TTOCN.FGP is set before the gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there is no gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = 1 in the reference message is ignored, as well as the event-trigger input pin fdcan_evt and the bits FDCAN_TTOCN.NIG, FDCAN_TTOCN.FGP, and FDCAN_TTOCN.TMG.

72.4.12 Stop watch

The stop watch function enables capturing of FDCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first must define local time, cycle time, or global time as the stop watch source via FDCAN_TTOCN.SWS. When FDCAN_TTOCN.SWS is different from 00 and TT interrupt register flag FDCAN_TTIR.SWE is 0, the actual value of the time selected by FDCAN_TTTOCN.SWS is copied into FDCAN_TTCPT.SWV on the next rising / falling edge (as configured via FDCAN_TTOCN.SWP) on stop watch trigger pin fdcan_swt. This sets interrupt flag FDCAN_TTIR.SWE. After the application program has read FDCAN_TTCPT.SWV, it may enable the next stop watch event by resetting FDCAN_TTIR.SWE to 0.

72.4.13 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN:

1. Level 1 only provides time-triggered operation using cycle time.
2. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

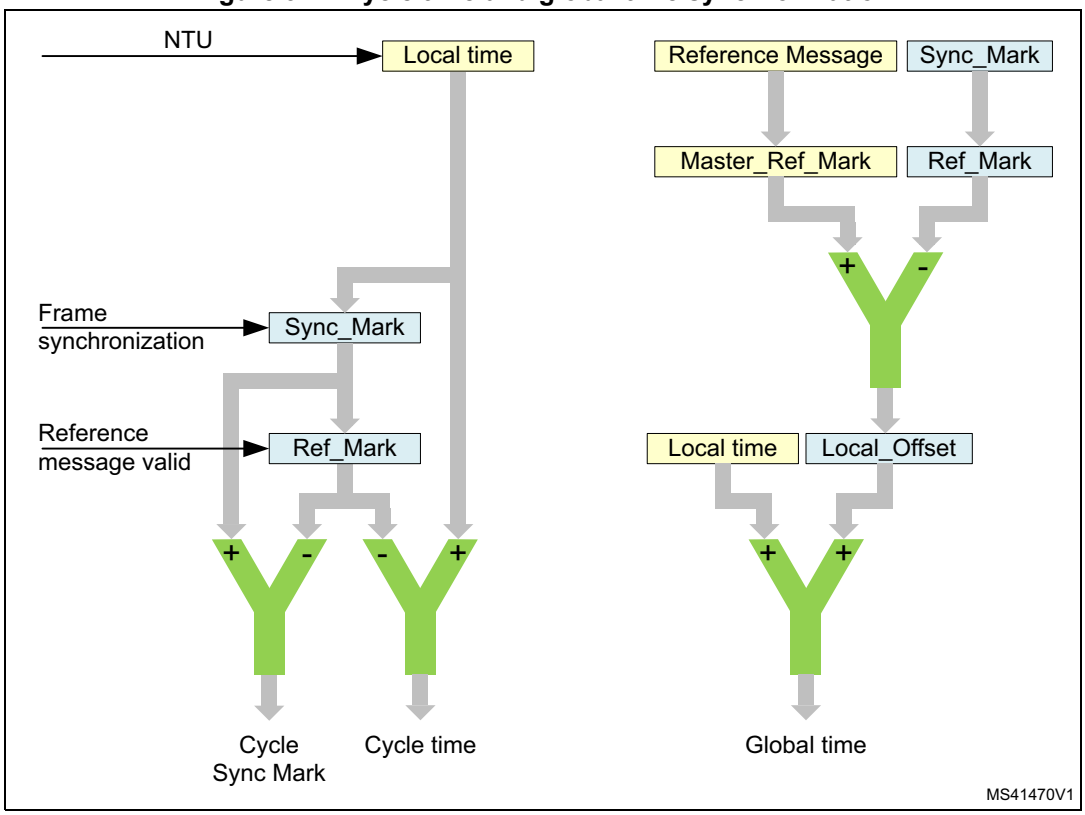
The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented eight times each NTU. If the length of the NTU is shorter than eight CAN clock periods (as may be configured in level 1, or as a result of clock calibration in level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only four times each NTU.

Figure 912 describes the synchronization of the cycle time and global time, performed in the same manner by all FDCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame

synchronization event. This frame synchronization event occurs at the sample point of each start of frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the cycle sync mark (cycle sync mark = Sync_Mark - Ref_Mark) stored in register FDCAN_TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (cycle time = local time - Ref_Mark).

Figure 912. Cycle time and global time synchronization



The cycle time that can be read from FDCAN_TTCTC.CT is the difference of the node local time and Ref_Mark, both synchronized into the APB clock domain and truncated to 16 bit.

The global time exists for TTCAN level 0 and level 2 only, in level 1 it is invalid. The node view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master uses its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3 and 4). The global time that can be read from FDCAN_TTLGT.GT is the sum of the node local time and its local offset, both synchronized into the APB clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see *Figure 913*). The node view of the global time is local time plus local offset. In a potential time master that has never received another time master reference message, Local_Offset is 0. When a node becomes the current time master after first having received other reference messages,

Local_Offset is frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register FDCAN_TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

Figure 913. TTCAN level 0 and level 2 drift compensation

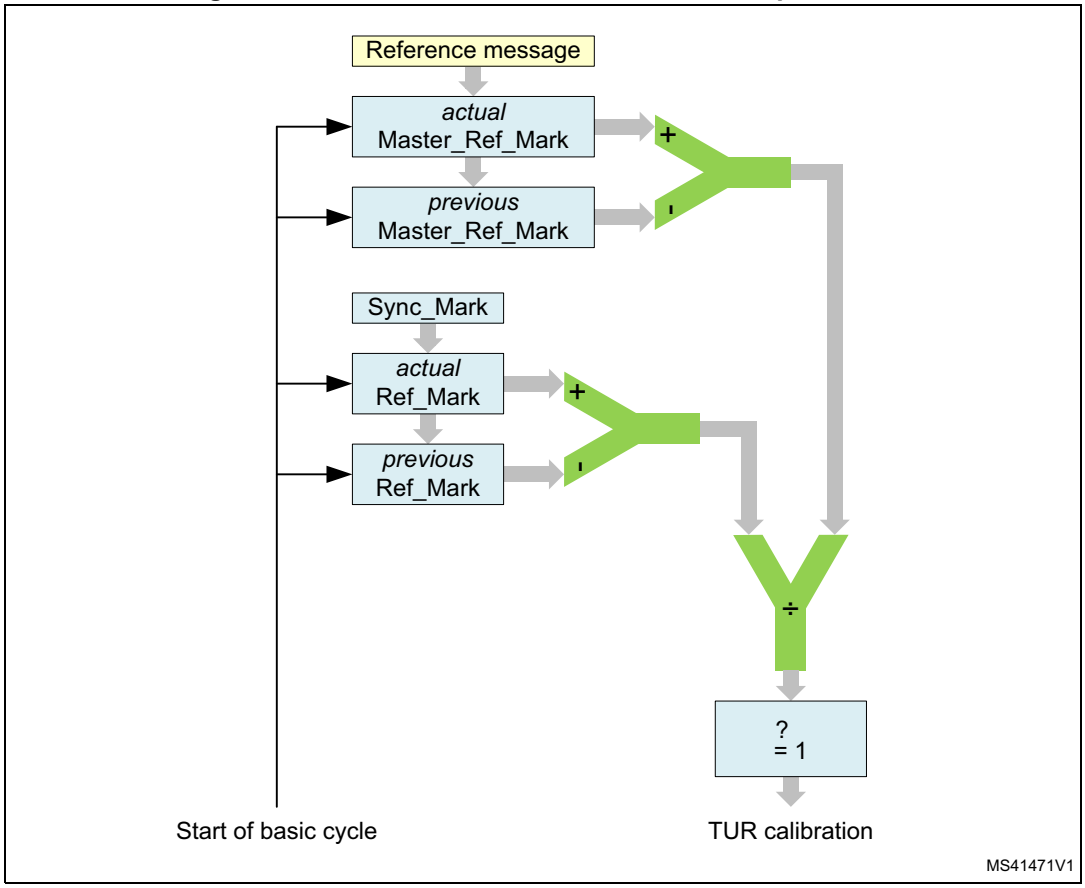


Figure 913 describes how in TTCAN levels 0 and 2 each time receiving node compensates the drift between its own local clock and the time master clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for FDCAN_TURNA.NAV is calculated. If the synchronization deviation SD = | NC - FDCAN_TURNA.NAV | ≤ SDL (synchronization deviation limit), the new value for FDCAN_TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN level 0 and level 2, FDCAN_TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN level 1, FDCAN_TOST.QCS is always 1.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit FDCAN_TTOCF.EECS.

The stop watch function (see *Section 72.4.12*) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated numerator configuration low to

FDCAN_TURCF.NCL (FDCAN_TURCF.DC cannot be updated during operation). The new value takes effect by writing FDCAN_TTOCN.ECS to 1.

The global time phase is adjusted by first writing the phase offset into the TT global time Preset register FDCAN_TTGTP. The new value takes effect by writing FDCAN_TTOCN.SGT to 1. The first reference message transmitted after the global time phase adjustment has the Disc_Bit set to 1.

FDCAN_TTOST.QGTP shows whether the node global time is in phase with the time master global time. FDCAN_TTOST.QGTP is permanently 0 in TTCAN level 1 and when the synchronization deviation limit is exceeded in TTCAN levels 0, 2 (FDCAN_TTOST.QCS = 0). It is temporarily 0 while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = 1 or when FDCAN_TTOST.QCS = 0.

72.4.14 TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

1. S0 - No error
2. S1 - Warning - Only notification of application, reaction application-specific.
3. S2 error - Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the reference trigger offset FDCAN_TTOST.RTO set to the maximum value of 127.
4. S3 - Severe error - Notification of application. All CAN bus operations are stopped, that is, transmission of dominant bits is not allowed, and FDCAN_CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set FDCAN_CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by FDCAN_TTIR.ELC. The error level is monitored by FDCAN_TTOST.EL.

The TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_error (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when the application failed to serve the application watchdog.The application watchdog is configured via FDCAN_TTOCF.AWL. It is served by reading register FDCAN_TTOST. When the watchdog is not served in time, bit FDCAN_TTOST.AWE and interrupt flag

FDCAN_TTIR.AW are set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode (FDCAN_CCCR.MON set to 1).

- CAN_Bus_Off (S3)
  - Entering CAN_Bus_Off state sets error level FDCAN_TTOST.EL to 11. CAN_Bus_Off state is signaled by FDCAN_PSR.BO = 1 and FDCAN_CCCR.INIT = 1.
- Scheduling_Error_2 (S2)
  - Sets error level FDCAN_TTOST.EL to 10 if the MSC of one Tx_Trigger has reached 7. In addition, the interrupt flag FDCAN_TTIR.SE2 is set. The error level FDCAN_TTOST.EL is reset to 00 at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.
- Tx_Overflow (S2)
  - Sets error level FDCAN_TTOST.EL to 10 when the Tx count is equal to or higher than the expected number of Tx_T riggers FDCAN_TTMLM.ENTT and a Tx_Trigger event occurs. In addition, the interrupt flag FDCAN_TTIR.TXO is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is no more than FDCAN_TTMLM.ENTT at the start of a new matrix cycle.
- Scheduling_Error_1 (S1)
  - Sets error level FDCAN_TTOST.EL to 01 if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than two, or if one of the MSCs of an exclusive Rx_Trigger has reached seven. In addition, the interrupt flag FDCAN_TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the error level FDCAN_TTOST.EL is reset to 00.
- Tx_Underflow (S1)
  - Sets error level FDCAN_TTOST.EL to 01 when the Tx count is less than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT at the start of a new matrix cycle. In addition, the interrupt flag FDCAN_TTIR.TXU is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is at least FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

72.4.15 TTCAN message handling

Reference message

For potential time masters the identifier of the reference message is configured via FDCAN_TTRMC.RID. No dedicated Tx buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte for TTCAN level 1 (that is, the first four data bytes for TTCAN level 0 and the first four data bytes for TTCAN level 2) is provided by the FSE.

In case the reference message Payload select FDCAN_TTRMC.RMPS is set, the rest of the reference message payload (level 1: bytes 2-8, levels 0, 2: bytes 5-6) is taken from Tx buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 750. Number of data bytes transmitted with a reference message

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4

Table 750. Number of data bytes transmitted with a reference message (continued)

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
1	0	4	1	4
1	1	4 + MBO	1 + MBO	4 + MBO

To send additional payload with the reference message in level 1 a DLC > 1 must be configured, for level 0 and level 2 a DLC > 4 is required. In addition, the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see *Table 750*). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the reference identifier FDCAN_TTRMC.RID is used.

Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see *Rx handler*).

The message status count MSC is part of the corresponding trigger memory element and must be initialized to 0 during configuration. It is updated while the TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has been completed. This must take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

Message transmission

For time-triggered message transmission the TTCAN supplies 32 dedicated Tx buffers (see *Transmit pause*). A Tx FIFO or Tx queue is not available when the FDCAN is configured for time-triggered operation (FDCAN_TTOCF.OM = 01 or 10).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program must update the data regularly and on time, synchronized to the cycle time. The user is responsible that no partially updated messages are transmitted. To assure this the user must proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has been completed by reading FDCAN_TXBTO
- Update the Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

Tx_Trigger_Continous

- Issue a cancellation request to reset the Tx buffer request pending bit
- Check whether the cancellation has finished by reading FDCAN_TXBCF
- Update Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

The message MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission cannot be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and cannot be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message may have been started within its transmit enable window but was not started because transmission was disabled (TTCAN in error level S2 or user has disabled this particular message).

The Tx buffers may be managed dynamically, that is, several messages with different identifiers may share the same Tx buffer element. In this case the user must ensure that no transmission request is pending for the Tx buffer element to be reconfigured by checking FDCAN_TXBRP.

If a Tx buffer with pending transmission request should be updated, the user must first issue a cancellation request and check whether the cancellation has been completed by reading FDCAN_TXBCF before it starts updating.

The Tx handler transfers a message from the message RAM to its intermediate output buffer at the trigger element, which becomes active immediately before the Tx_Trigger element (which defines the beginning of the transmit window). During and after the transfer time the transmit message may not be updated and its FDCAN_TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger must be large enough to guarantee that the Tx handler can read four words from the message RAM even at high RAM access load from other modules.

Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx buffer request pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx buffer request pending remains set. When the transmission was not successful due to disturbances, it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message must arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission is not started if the corresponding transmission request pending flag is reset by a successful Tx cancellation.

In single transmit windows, the Tx handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmission is attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the user), the fourth request is ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window are started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This must be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx buffer needs to be updated.

72.4.16 TTCAN interrupt and error handling

The TT interrupt register FDCAN_TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT interrupt enable register FDCAN_TTIE. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the FDCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, for example by disturbances on the CAN bus, they are handled by the FDCAN protocol failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (levels 0, 2 only) that require a reaction by the application program. With a stop watch event triggered by a rising edge on pin fdcan_swt

internal time values are captured. The trigger time mark interrupt notifies the application that a specific Time_Base_Trigger is reached. The register time mark interrupt signals that the time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to the time mark FDCAN_TTTMK.TM. It can also be used to finish a gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

72.4.17 Level 0

TTCAN level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. In this mode the FDCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of FDCAN_TTOCF.GEN is ignored. External event-synchronized operation is not available in level 0. A synchronized time base is maintained by the transmission of reference messages.

In level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (FDCAN_TTIR.TTMI) is set when the cycle time has reached FDCAN_TTOCF.IRTO ´ 0x200, it reminds the user to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (FDCAN_TTIR.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (FDCAN_TTIR.RTMI) are also possible.

The reference message is configured as for level 2 operation. Received reference messages are recognized by the identifier configured in register FDCAN_TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the user sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- FDCAN_TTRMC
- FDCAN_TTOCF except EVTP, AWL, GEN
- FDCAN_TTMLM except ENTT, TXEW
- FDCAN_TURCF

Level 0 operation is controlled via:

- FDCAN_TTOCN except NIG, TMG, FGP, GCS, TTMIE
- FDCAN_TTGTP
- FDCAN_TTTMK
- FDCAN_TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- FDCAN_TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time FDCAN_TTOCF.IRTO 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

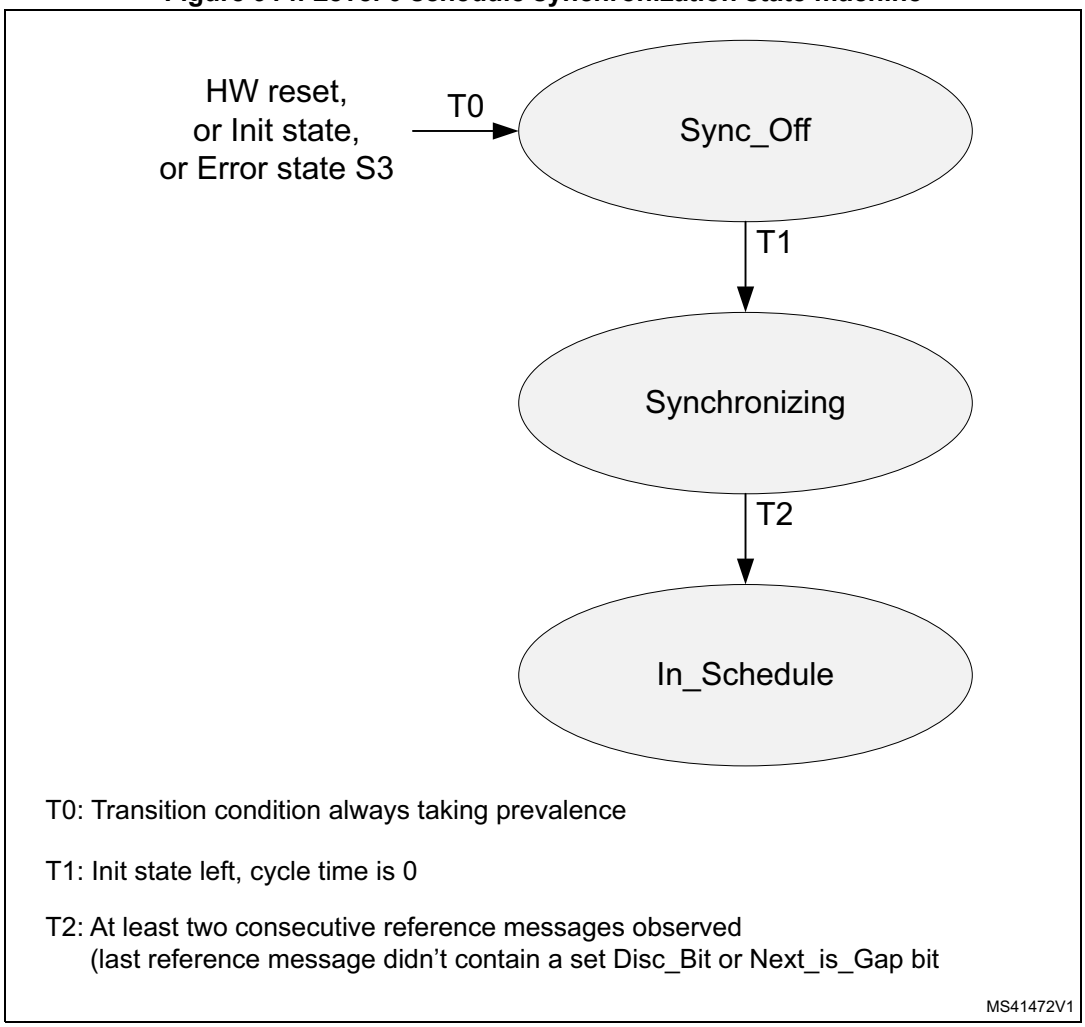
Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

Synchronizing

Figure 914 describes the states and the state transitions in TTCAN level 0 operation. level 0 has no In_Gap state.

Figure 914. Level 0 schedule synchronization state machine



Handling of error levels

During level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

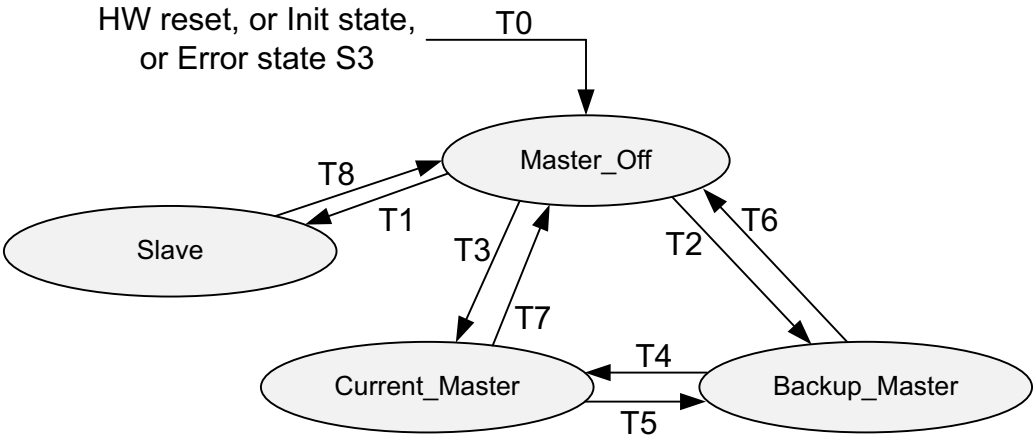
Since no S1 and S2 errors are possible, the error level can only switch between S0 (No error) and S3 (Severe error). In TTCAN level 0 an S3 error is handled differently. When error level S3 is reached, both FDCAN_TTOST.SYS and FDCAN_TTOST.MS are reset, and interrupt flags FDCAN_TTIR.GTE and FDCAN_TTIR.GTD are set.

When error level S3 (FDCAN_TTOST.EL = 11) is entered, bus monitoring mode is (contrary to TTCAN level 1 and level 2) not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

Master slave relation

Figure 915 describes the master slave relation in TTCAN level 0. In case of an S3 error the FDCAN returns to state Master_Off.

Figure 915. Level 0 master to slave relation



T0: Transition condition, always taking prevalence

T1: Reference message observed when not potential time master

T2: Reference message observed with master priority different from own master priority, Error state ≠ S3

T3: Reference message observed with master priority equal to own master priority, Error state ≠ S3

T4: Reference message observed with own master priority

T5: Reference message observed with master priority higher than own master priority

T6: Error state S3

T7: Error state S3

T8: Error state S3

MS41473V1

72.4.18 Synchronization to external time schedule

This feature can be used to synchronize the phase of the FDCAN schedule to an external schedule (for example that of a second TTCAN network or FlexRay network). It is applicable only when the FDCAN is current time master (FDCAN_TTOST.MS = 11).

External synchronization is controlled by event trigger input pin fdcan_evt. If bit FDCAN_TTOCN.ESCN is set, a rising edge at event trigger pin fdcan_evt the FDCAN compares its actual cycle time with the target phase value configured by FDCAN_TTGTP.CTP.

Before setting FDCAN_TTOCN.ESCN the user must adapt the phases of the two time schedules for example by using the FDCAN gap control (see *Section 72.4.11*). When the user sets FDCAN_TTOCN.ESCN, FDCAN_TTOSTSPL is set.

If the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is greater than 9 NTU, the phase lock bit FDCAN_TTOST.SPL is reset, and interrupt flag FDCAN_TTIR.CSM is set.

FDCAN_TTOST.SPL is also reset (and FDCAN_TTIR.CSM is set), when another node becomes time master.

If both FDCAN_TTOST.SPL and FDCAN_TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is lower or equal to nine NTU, the phase lock bit FDCAN_TTOST.SPL remains set, and the measured difference is used as the reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin fdcan_evt is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with FDCAN_*TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.*

72.4.19 FDCAN Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in *Table 751*, the description is provided in *Table 752*.

Table 751. Rx buffer and FIFO element

Bit	31			24	23			16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]								
R1	ANMF	FIDX[6:0]			Res.	FDF	BRS	DLC[3:0]	RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮				⋮			
Rn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]		DBm-3[7:0]	

The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_RXESC.

Table 752. Rx buffer and FIFO element description

Field	Description
R0 bit 31 ESI	Error state indicator 0: Transmitting node is error active 1: Transmitting node is error passive
R0 bit 30 XTD	Extended identifier Signals to the user whether the received frame has a standard or extended identifier. 0: 11-bit standard identifier 1: 29-bit extended identifier
R0 bit 29 RTR	Remote transmission request Signals to the user whether the received frame is a data frame or a remote frame. 0: Received frame is a data frame 1: Received frame is a remote frame

Table 752. Rx buffer and FIFO element description (continued)

Field	Description
R0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames may be enabled via FDCAN_GFC.ANFS and FDCAN_GFC.ANFE. 0: Received frame matching filter index FIDX 1: Received frame did not match any Rx filter element
R1 bits 30:24 FIDX[6:0]	Filter index 0-127 = index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to FDCAN_SIDFC.LSS. - 1 or FDCAN_XIDFC.LSE. - 1.
R1 bit 21 FDF	FD Format 0: Standard frame format 1: FDCAN frame format (new DLC-coding and CRC)
R1 bit 20 BRS	Bitrate Switch 0: Frame received without bitrate switching 1: Frame received with bitrate switching
R1 bits 19:16 DLC[3:0]	Data length code 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 15:0 RXTS[15:0]	Rx timestamp Timestamp counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.
R2 bits 31:24 DB3[7:0]	Data Byte 3
R2 bits 23:16 DB2[7:0]	Data Byte 2
R2 bits 15:8 DB1[7:0]	Data Byte 1
R2 bits 7:0 DB0[7:0]	Data Byte 0
R3 bits 31:24 DB7[7:0]	Data Byte 7
R3 bits 23:16 DB6[7:0]	Data Byte 6
R3 bits 15:8 DB5[7:0]	Data Byte 5
R3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮

Table 752. Rx buffer and FIFO element description (continued)

Field	Description
Rn bits 31:24 DBm[7:0]	Data Byte m
Rn bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn bits 7:0 DBm-3[7:0]	Data Byte m-3

72.4.20 FDCAN Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration FDCAN_TXBC.TFQS and FDCAN_TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_TXESC.

Table 753. Tx buffer and FIFO element

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T0	ESI	XTD	RTR	ID[28:0]																												
T1	MM[7:0]								EFC	Res.	FDF	BPS	DLC[3:0]				Reserved															
T2	DB3[7:0]								DB2[7:0]								DB1[7:0]								DB0[7:0]							
T3	DB7[7:0]								DB6[7:0]								DB5[7:0]								DB4[7:0]							
⋮	⋮								⋮								⋮															
Tn	DBm[7:0]								DBm-1[7:0]								DBm-2[7:0]								DBm-3[7:0]							

Table 754. Tx buffer element description

Field	Description
T0 bit 31 ESI(1)	Error state indicator 0: ESI bit in CAN FD format depends only on error passive flag 1: ESI bit in CAN FD format transmitted recessive
T0 bit 30 XTD	Extended identifier 0: 11-bit standard identifier 1: 29-bit extended identifier
T0 bit 29 RTR(2)	Remote transmission request 0: Transmit data frame 1: Transmit remote frame

Table 754. Tx buffer element description (continued)

Field	Description
T0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
T1 bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 bit 23 EFC	Event FIFO control 0: Don’t store Tx events 1: Store Tx events
T1 bit 21 FDF	FD Format 0: Frame transmitted in classic CAN format 1: Frame transmitted in CAN FD format
T1 bit 20 BRS(3)	Bitrate switching 0: CAN FD frames transmitted without bitrate switching 1: CAN FD frames transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 bits 31:24 DB3[7:0]	Data Byte 3
T2 bits 23:16 DB2[7:0]	Data Byte 2
T2 bits 15:8 DB1[7:0]	Data Byte 1
T2 bits 7:0 DB0[7:0]	Data Byte 0
T3 bits 31:24 DB7[7:0]	Data Byte 7
T3 bits 23:16 DB6[7:0]	Data Byte 6
T3 bits 15:8 DB5[7:0]	Data Byte 5
T3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Tn bits 31:24 DBm[7:0]	Data Byte m
Tn bits 23:16 DBm-1[7:0]	Data Byte m-1

Table 754. Tx buffer element description (continued)

Field	Description
Tn bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn bits 7:0 DBm-3[7:0]	Data Byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmit the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if FDCAN_CCCR.FDOE enables the transmission in CAN FD format.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled FDCAN_CCCR.FDOE = 1. Bit BRS is only evaluated when in addition FDCAN_CCCR.BRSE = 1.

72.4.21 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the user gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register FDCAN_TXEFS.

Table 755. Tx Event FIFO element

Bit	31			24	23					16	15	8	7	0
E0	ESI	XTD	RTR		ID[28:0]									
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]		TXTS[15:0]			

Table 756. Tx Event FIFO element description

Field	Description
E0 bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 bit 30 XTD	Extended Identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
E1 bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

Table 756. Tx Event FIFO element description (continued)

Field	Description
E1 bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 bit 20 BRS	Bitrate switching – 0: Frame transmitted without bitrate switching – 1: Frame transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code – 0 - 8: Frame with 0-8 data bytes transmitted – 9 - 15: Frame with eight data bytes transmitted
E1 bits 15:0 TXTS[15:0]	Tx timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.

72.4.22 FDCAN standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FDCAN_SIDFC.FLSSA plus the index of the filter element (0 … 127).

Table 757. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 758. Standard message ID filter element field description

Field		Description
Bit 31:30 SFT[1:0](1)		Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]		Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer or as debug message, configuration of FDCAN_SFT[1:0] ignored
Bits 26:16 SFID1[10:0]		Standard filter ID 1 First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15:0	SFID2[15:10]	Standard filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: – SFEC = 001 ... 110 Second ID of standard ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	SFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. SFID2[8] is used by the calibration unit.
	SFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

1. With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = “000”).

Note: In case a reserved value is configured, the filter element is considered disabled.

72.4.23 FDCAN extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the filter list extended start address FDCAN_XIDFC.FLESA plus two times the index of the filter element (0 … 63).

Table 759. Extended message ID filter element

Bit	31	30	29	28	0
F0	EFEC[2:0]			EFID1[28:0]	
F1	EFT[1:0]		Reserved	EFID2[28:0]	

Table 760. Extended message ID filter element field description

Field	Description
F0 bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer, configuration of EFT[1:0] ignored
F0 bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only FDCAN_XIDAM masking mechanism.
F1 bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), FDCAN_XIDAM mask not applied

Table 760. Extended message ID filter element field description (continued)

Field		Description
F1 bits 28:0	EFID2[10:0]	Extended filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – SFEC = 001 ... 110 Second ID of extended ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. EFID2[8] interface is used by the calibration unit.
	EFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

72.4.24 FDCAN trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a trigger memory element, its address is the trigger memory start address FDCAN_TTTMC.TMSA plus the index of the trigger memory element (0 … 63).

Table 761. Trigger memory element

Bit	31 24	23 16	15 8	7 0
T0	TM[15:0]		Res. CC[6:0]	Res. TMIN TMEX TYPE[3:0]
T1	Res.	FTYPE MNR[6:0]	Res.	MSC[2:0]

Table 762. Trigger memory element description

Field	Description
T0 bits 31:16 TM[15:0]	Time mark Cycle time for which the trigger becomes active.
T0 bit 14:8 CC[6:0]	Cycle code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. – 0b000000x valid for all cycles – 0b000001c valid every 2^{nd} cycle at cycle count mod2 = c – 0b00001cc valid every 4^{th} cycle at cycle count mod4 = cc – 0b0001ccc valid every 8^{th} cycle at cycle count mod8 = ccc – 0b001cccc valid every 16^{th} cycle at cycle count mod16 = cccc – 0b01ccccc valid every 32^{nd} cycle at cycle count mod32 = ccccc – 0b1cccccc valid every 64^{th} cycle at cycle count mod64 = cccccc

Table 762. Trigger memory element description (continued)

Field	Description
T0 bit 5 TMIN	Time mark event internal – 0: No action – 1: FDCAN_TTIR.TTMI is set when trigger memory element becomes active
T0 bit 4 TMEX	Time mark event external – 0: No action – 1: Pulse at output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance with the length of one period is generated when the time arc of the trigger memory element becomes active and FDCAN_TTOCN.TTMIE = 1
T0 bit 3:0 TYPE[3:0]	Trigger type – 0000 Tx_Ref_Trigger - valid when not in gap – 0001 Tx_Ref_Trigger_Gap - valid when in gap – 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window – 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window – 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window – 0101 Tx_Trigger_Merged - starts a merged arbitration window – 0110 Watch_Trigger - valid when not in gap – 0111 Watch_Trigger_Gap - valid when in gap – 1000 Rx_Trigger - check for reception – 1001 Time_Base_Trigger - only control TMIN, TMEX – 1010 … 1111=End_of_List - illegal type, causes configuration error
T1 bit 23FTYPE	Filter type – 0: 11-bit standard message ID – 1: 29-bit extended message ID
T1 bit 22:16 MNR[6:0](1)	Message number – Transmission: trigger is valid for configured Tx buffer number. Valid values are 0 to 31. – Reception: trigger is valid for standard/extended message ID filter element number. Valid values are, respectively 0 to 63 and 0 to 127.
T1 bits 2:0 MSC[2:0]	Message status count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). – 0-7= Actual status

1. The trigger memory elements must be written when the FDCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual reference trigger offset (TTOST[RTO]).

72.5 FDCAN registers

72.5.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 3

Bits 27:24 **STEP[3:0]**: Step of core release = 2

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day =18

72.5.2 FDCAN Endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

72.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is dedicated to data bit timing phase and only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN time quantum may be programmed in the range from 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock periods.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (DTSEG1 + DTSEG2 + 3) tq for programmed values, or (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq for functional values.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bitrate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of these quanta. Valid values for the baudrate prescaler are 0 to 31. The hardware interpreters this value as the programmed value plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS1} = (DTSEG1 + 1) x tq.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS2} = (DTSEG2 + 1) x tq.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (DSJW + 1) x tq.

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x00000A33 configures the FDCAN for a bit rate of 500 kbit/s.

Note: The data phase bit rate must be higher than or equal to the nominal bit rate.

72.5.4 FDCAN test register (FDCAN_TEST)

Write access to this register must be enabled by setting bit FDCAN_CCCR.TEST to 1. All register functions are set to their reset values when bit FDCAN_CCCR.TEST is reset.

Loop back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

Monitors the actual value of transmit pin FDCANx_RX. This bit is a synchronized version of the FDCANx_RX, it goes back to 1 after reset if connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop back mode

0: Reset value, loop back mode is disabled

1: Loop back mode is enabled (see *Test modes*)

Bits 3:0 Reserved, must be kept at reset value.

72.5.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the FDCAN_RWD.WDC bits.

The counter is reloaded with FDCAN_RWD.WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops and the interrupt flag FDCAN_IR.WDI bit is set. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00 the counter is disabled.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.5.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**:
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bitrate switching
0: Bitrate switching for transmissions disabled
1: Bitrate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, register TEST holds reset values
1: Test mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
Bit MON can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the user at any time.
0: Bus monitoring mode is disabled
1: Bus monitoring mode is enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop is requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN may be set in power down by stopping APB clock and kernel clock

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bitrates. The application tests different bitrates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time. This bit is set automatically set to 1 when the Tx handler was not able to read data from the message RAM in time. If the FDCAN is connected to a clock calibration on CAN unit, ASM bit is set by hardware as long as the calibration is not completed.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers
1: The CPU has write access to the protected configuration registers (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization is started (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer must ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

72.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is dedicated to the nominal bit timing used during the arbitration phase, and is only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = (BRP + 1) FDCAN clock period fdcan_tq_ck

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Should be smaller than NTSEG2, valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (NSJW + 1) x tq.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]**: Bitrate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The value used by the hardware is the one programmed, incremented by 1.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The value used by the hardware is the one programmed, incremented by 1 (t_{BS1} = (NTSEG1 + 1) x tq).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1 (t_{BS2} = (NTSEG2 + 1) x tq).

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x06000A03 configures the FDCAN for a bit rate of 500 kbit/s.

72.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value used (tim3_cnt[0:15])

11: Same as 00.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.5.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When FDCAN_TSCC.TSS = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. A wrap around sets interrupt flag FDCAN_IR.TSW. Write access resets the counter to 0. When FDCAN_TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 not caused by write access to FDCAN_TSCV.

72.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in Continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.

00: Continuous operation

01: Timeout controlled by Tx event FIFO

10: Timeout controlled by Rx FIFO 0

11: Timeout controlled by Rx FIFO 1

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **ETOC**: Enable timeout counter

0: Timeout counter disabled

1: Timeout counter enabled

This is a write-protected bit, write access is possible only when the bit 1 (CCE) and bit 0 (INIT) of FDCAN_CCCR register are set to 1.

For more details see *Timeout counter*.

72.5.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. When decremented to 0, interrupt flag FDCAN_IR.TOO is set and the timeout counter is stopped. Start and reset/restart conditions are configured via FDCAN_TOCC.TOS.

72.5.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag FDCAN_IR.ELO.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128

1: The receive error counter has reached the error passive level of 128

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When FDCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

72.5.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs_r	rs_r	rs_r	r	r	r	r	r	rs_r	rs_r	rs_r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and FDCAN_TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access
1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.
0: Since this bit was reset by the CPU, no FDCAN message has been received
1: Message in FDCAN format with EDL flag set has been received
Access type is RX: reset on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not have its BRS flag set
1: Last received FDCAN message had its BRS flag set
Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not ha ve its ESI flag set
1: Last received FDCAN message had its ESI flag set
Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of an FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to 0 when an FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status

0: The FDCAN is not Bus_Off
1: The FDCAN is in Bus_Off state

Bit 6 **EW**: Warning status
0: Both error counters are below the Error_Warning limit of 96
1: At least one of error counter has reached the Error_Warning limit of 96

Bit 5 **EP**: Error passive
0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected
1: The FDCAN is in the Error_Passive state

Bits 4:3 **ACT[1:0]**: Activity
Monitors the module CAN communication state.
00: Synchronizing: node is synchronizing on CAN communication
01: Idle: node is neither receiver nor transmitter
10: Receiver: node is operating as receiver
11: Transmitter: node is operating as transmitter

Bits 2:0 **LEC[2:0]:** Last error code
The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error.
000: No error: No error occurred since LEC has been reset by successful reception or transmission.
001: Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010: Form error: A fixed format part of a received frame has the wrong format.
011: AckError: The message transmitted by the FDCAN was not acknowledged by another node.
100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
111: NoChange: Any read access to the protocol status register re-initializes the LEC to ‘7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.
Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in FLEC instead of LEC. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a Form error, not Stuff error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting FDCAN_CCCR.INIT. If the device goes Bus_Off, it sets FDCAN_CCCR.INIT of its own, stopping all bus activities. Once FDCAN_CCCR.INIT has been cleared by the CPU, the device waits for 129 occurrences of bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the error management counters are reset. During the waiting time after the reset of FDCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 error code is written to FDCAN_PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. FDCAN_ECR.REC is used to count these sequences.

72.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.

These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.5.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 28 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 27 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 26 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 25 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 24 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 23 **EP**: Error passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 22 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow
1: Overflow of CAN error logging counter occurred

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRX**: Message stored to dedicated Rx buffer
The flag is set whenever a received message has been stored into a dedicated Rx buffer.
0: No Rx buffer updated
1: At least one received message stored into a Rx buffer

Bit 18 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 17 **MRAF**: Message RAM access failure
The flag is set when the Rx handler
- Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- Was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated or the New data flag for a dedicated Rx buffer is not set. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the user must reset FDCAN_CCCR.ASM.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 16 **TSW**: Timestamp wraparound
0: No timestamp counter wraparound
1: Timestamp counter wraparound

Bit 15 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0

Bit 14 **TEFF**: Tx event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bit 13 **TEFW**: Tx event FIFO watermark reached
0: Tx event FIFO fill level below watermark
1: Tx event FIFO fill level reached watermark

Bit 12 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element

Bit 11 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 10 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 9 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 8 **HPM**: High priority message
0: No high priority message received
1: High priority message received

Bit 7 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 6 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 5 **RF1W**: Rx FIFO 1 watermark reached
0: Rx FIFO 1 fill level below watermark
1: Rx FIFO 1 fill level reached watermark

Bit 4 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 3 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 2 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 1 **RF0W**: Rx FIFO 0 watermark reached
0: Rx FIFO 0 fill level below watermark
1: Rx FIFO 0 fill level reached watermark

Bit 0 **RF0N**: Rx FIFO 0 New message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

72.5.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register is signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAE**: Access to Reserved address enable

Bit 28 **PEDE**: Protocol error in data phase enable

Bit 27 **PEAE**: Protocol error in Arbitration phase enable

Bit 26 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 25 **BOE**: Bus_Off status
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXE**: Message stored to dedicated Rx buffer interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TEFWE**: Tx event FIFO watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **HPME**: High priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1WE**: Rx FIFO 1 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0WE**: Rx FIFO 0 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

72.5.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific interrupt flag from the interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAL**: Access to reserved address line

Bit 28 **PEDL**: Protocol error in data phase line

Bit 27 **PEAL**: Protocol error in arbitration phase line

Bit 26 **WDIL**: Watchdog interrupt line

Bit 25 **BOL**: Bus_Off status

Bit 24 **EWL**: Warning status interrupt line

Bit 23 **EPL**: Error passive interrupt line

Bit 22 **ELOL**: Error logging overflow interrupt line

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXL**: Message stored to dedicated Rx buffer interrupt line

Bit 18 **TOOL**: Timeout occurred interrupt line

Bit 17 **MRAFL**: Message RAM access failure interrupt line

Bit 16 **TSWL**: Timestamp wraparound interrupt line

Bit 15 **TEFLL**: Tx event FIFO element Lost interrupt line

Bit 14 **TEFFL**: Tx event FIFO full interrupt line

Bit 13 **TEFWL**: Tx event FIFO watermark reached interrupt line

Bit 12 **TEFNL**: Tx event FIFO new entry interrupt line

Bit 11 **TFEL**: Tx FIFO empty interrupt line

Bit 10 **TCFL**: Transmission cancellation finished interrupt line

Bit 9 **TCL**: Transmission completed interrupt line

Bit 8 **HPML**: High priority message interrupt line

Bit 7 **RF1LL**: Rx FIFO 1 message lost interrupt line

Bit 6 **RF1FL**: Rx FIFO 1 full interrupt line

Bit 5 **RF1WL**: Rx FIFO 1 watermark reached interrupt line

Bit 4 **RF1NL**: Rx FIFO 1 new message interrupt line

Bit 3 **RF0LL**: Rx FIFO 0 message lost interrupt line

Bit 2 **RF0FL**: Rx FIFO 0 full interrupt line

Bit 1 **RF0WL**: Rx FIFO 0 watermark reached interrupt line

Bit 0 **RF0NL**: Rx FIFO 0 new message interrupt line

72.5.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1

0: Interrupt line fdcan_intr1_it disabled

1: Interrupt line fdcan_intr1_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

72.5.19 FDCAN global filter configuration register (FDCAN_GFC)

Global settings for message ID filtering. The global filter configuration register controls the filter path for standard and extended messages as described in *Figure 905* and *Figure 906*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept non-matching frames standard
Defines how received messages with 11-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard ID
1: Reject all remote frames with 11-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard ID
1: Reject all remote frames with 29-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC)

Settings for 11-bit standard message ID filtering.The standard ID filter configuration register controls the filter path for standard messages as described in *Figure 905*.

Address offset: 0x0084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLSSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSS[7:0]**: List size standard
0: No standard message ID filter
1-128: Number of standard message ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLSSA[13:0]**: Filter list standard start address
Start address of standard message ID filter list (32-bit word address, see *Table 757*).These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC)

Settings for 29-bit extended message ID filtering. The FDCAN extended ID filter configuration register controls the filter path for standard messages as described in *Figure 906*.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLESA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSE[7:0]**: List size extended
0: No extended message ID filter
1-128: Number of extended ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLESA[13:0]**: Filter list extended start address
Start address of extended message ID filter list (32-bit word address, see *Table 759: Extended message ID filter element*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0090

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID Mask
For acceptance filtering of extended frames the extended ID AND Mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.5.23 FDCAN high priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element.
0: Standard filter list
1: Extended filter list

Bits 14:8 **FIDX[6:0]**: Filter index
Index of matching filter element. Range is 0 to FDCAN_SIDFC[LSS] - 1 or FDCAN_XIDFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:0 **BIDX[5:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

72.5.24 FDCAN new data 1 register (FDCAN_NDAT1)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[31:0]**: New data[31:0]

The register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

72.5.25 FDCAN new data 2 register (FDCAN_NDAT2)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[63:32]**: New data[63:32]

The register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears clear the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

72.5.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)

Address offset: 0x00A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM[6:0]							Res.	F0S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F0OM**: FIFO 0 operation mode

FIFO 0 can be operated in blocking or in overwrite mode.

0: FIFO 0 blocking mode

1: FIFO 0 overwrite mode

Bits 30:24 **F0WM[6:0]**: FIFO 0 watermark
0: Watermark interrupt disabled
1-64: Level for Rx FIFO 0 watermark interrupt (FDCAN_IR.RF0W)
>64: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F0S[6:0]**: Rx FIFO 0 size
0: No Rx FIFO 0
1-64: Number of Rx FIFO 0 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 0 elements are indexed from 0 to F0S-1.

Bits 15:2 **F0SA[13:0]**: Rx FIFO 0 start address
Start address of Rx FIFO 0 in message RAM (32-bit word address, see *Figure 904*).

Bits 1:0 Reserved, must be kept at reset value.

72.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x00A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	F0PI[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of interrupt flag FDCAN_IR.RF0L. When FDCAN_IR.RF0L is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F0PI[5:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F0GI[5:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F0FL**[6:0]: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 64.

72.5.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F0AI[5:0]**: Rx FIFO 0 acknowledge index
After the user has read a message or a sequence of messages from Rx FIFO 0 it must write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 get index FDCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 fill level FDCAN_RXF0S.F0FL.

72.5.29 FDCAN Rx buffer configuration register (FDCAN_RXBC)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **RBSA[13:0]**: Rx buffer start address
Configures the start address of the Rx buffers section in the message RAM (32-bit word address). Also used to reference debug messages A, B, C.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)

Address offset: 0x00B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F1OM	F1WM[6:0]							Res.	F1S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F1SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F1OM**: FIFO 1 operation mode

FIFO 1 can be operated in blocking or in overwrite mode.

0: FIFO 1 blocking mode

1: FIFO 1 overwrite mode

Bits 30:24 **F1WM[6:0]**: Rx FIFO 1 watermark

0: Watermark interrupt disabled

1-64: Level for Rx FIFO 1 watermark interrupt (FDCAN_IR.RF1W)

>64: Watermark interrupt disabled.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F1S[6:0]**: Rx FIFO 1 size

0: No Rx FIFO 1

1-64: Number of Rx FIFO 1 elements

>64: Values greater than 64 are interpreted as 64

The Rx FIFO 1 elements are indexed from 0 to F1S - 1.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **F1SA[13:0]**: Rx FIFO 1 start address

start address of Rx FIFO 1 in message RAM (32-bit word address, see *Figure 904*).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x00B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	F1PI[5:0]					
r	r					r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
		r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:30 **DMS[1:0]**: Debug message status
00: Idle state, wait for reception of debug messages
01: Debug message A received
10: Debug messages A, B received
11: Debug messages A, B, C received

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost
This bit is a copy of interrupt flag FDCAN_IR.RF1L. When FDCAN_IR.RF1L is reset, this bit is also reset.
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0.

Bit 24 **F1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F1PI[5:0]**: Rx FIFO 1 put index
Rx FIFO 1 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F1GI[5:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F1FL[6:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1, range 0 to 64

72.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x00B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F1AI[5:0]**: Rx FIFO 1 acknowledge index

After the user has read a message or a sequence of messages from Rx FIFO 1 it must write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 get index FDCAN_RXF1S.F1GI. to F1AI + 1 and update the FIFO 1 fill level FDCAN_RXF1S.F1FL.

72.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC)

Configures the number of data bytes belonging to an Rx buffer / Rx FIFO element. Data field sizes higher than 8 bytes are intended for CAN FD operation only.

Address offset: 0x00BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RBDS[2:0]			Res.	F1DS[2:0]			Res.	F0DS[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **RBDS[2:0]**: Rx buffer data field size

000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **F1DS[2:0]**: Rx FIFO 0 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **F0DS[2:0]**: Rx FIFO 1 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

72.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 29:24 **TFQS[5:0]**: Transmit FIFO/queue size
0: No Tx FIFO/queue
1-32: Number of Tx buffers used for Tx FIFO/queue
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **NDTB[5:0]**: Number of dedicated transmit buffers
0: No dedicated Tx buffers
1-32: Number of dedicated Tx buffers
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TBSA[13:0]**: Tx buffers start address
Start address of Tx buffers section in message RAM (32-bit word address, see *Figure 904*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

Note: The sum of TFQS and NDTB cannot be larger than 32. There is no check for erroneous configurations. The Tx buffers section in the message RAM starts with the dedicated Tx buffers.

72.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in register FDCAN_TXBRP. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]				
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0 Tx FIFO/queue not full
1 Tx FIFO/queue full

Bits 20:16 **TFQPI[4:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 31

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **TFGI[4:0]**:
Tx FIFO get index.
Tx FIFO read index pointer, range 0 to 31. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TFFL[5:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1).

Note: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get index indicate the number of the Tx buffer starting with the first dedicated Tx buffers. For example: For a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

72.5.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC)

Configures the number of data bytes belonging to a Tx buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TBDS[2:0]**: Tx buffer data field size:
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

72.5.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TRP[31:0]**:Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via register FDCAN_TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register FDCAN_TXBCR.

FDCAN_TXBRP bits are set only for those Tx buffers configured via FDCAN_TXBC. After an FDCAN_TXBRP bit has been set, a Tx scan (see *Filtering for Debug messages*) is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of register FDCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding FDCAN_TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via FDCAN_TXBCF

after successful transmission together with the corresponding FDCAN_TXBTO bit

when the transmission has not yet been started at the point of cancellation

when the transmission has been aborted due to lost arbitration

when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately, the corresponding FDCAN_TXBRP bit is reset.

72.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AR[31:0]**:Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the user to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. FDCAN_TXBAR bits are set only for those Tx buffers configured via FDCAN_TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

72.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CR[31:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding cancellation request bit; writing a 0 has no impact.

This enables the user to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. FDCAN_TXBCR bits are set only for those Tx buffers configured via FDCAN_TXBC. The bits remain set until the corresponding FDCAN_TXBRP bit is reset.

0: No cancellation pending

1: Cancellation pending

72.5.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TO[31:0]**: Transmission occurred

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

72.5.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CF[31:0]**: Cancellation finished

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

72.5.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIE[31:0]**: Transmission interrupt enable

Each Tx buffer has its own transmission interrupt enable bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

72.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFIE[31:0]**: Cancellation finished interrupt enable
Each Tx buffer has its own cancellation finished interrupt enable bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

72.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)

Address offset: 0x00F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EFWM[5:0]**: Event FIFO watermark
0: Watermark interrupt disabled
1-32: Level for Tx event FIFO watermark interrupt (FDCAN_IR.TEFW)
>32: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **EFS[5:0]**: Event FIFO size.
0: Tx event FIFO disabled
1-32: Number of Tx event FIFO elements
>32: Values greater than 32 are interpreted as 32
The Tx event FIFO elements are indexed from 0 to EFS-1.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **EFSA[13:0]**: Event FIFO start address
Start address of Tx event FIFO in message RAM (32-bit word address, see *Figure 904*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	EFPI[4:0]				
						r	r				r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of interrupt flag FDCAN_IR.TEFL. When FDCAN_IR.TEFL is reset, this bit is also reset.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **EFPI[4:0]**: Event FIFO put index
Tx event FIFO write index pointer, range 0 to 31.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **EFGI[4:0]**: Event FIFO get index
Tx event FIFO read index pointer, range 0 to 31.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **EFFL[5:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO, range 0 to 31.

72.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EFAI[4:0]**: Event FIFO acknowledge index

After the user has read an element or a sequence of elements from the Tx event FIFO, it must write the index of the last element read from Tx event FIFO to EFAI. This sets the Tx event FIFO get index FDCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO 0 fill level FDCAN_TXEFS.EFFL.

72.5.47 FDCAN register map

Table 763. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2 [3:0]				DSJW[3:0]			
	Reset value									0			0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX [1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
	Reset value																	0	0	0	0			0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]							
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0		0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS [1:0]	
	Reset value													0	0	0	0															0	0

Table 763. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS [1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1														0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003F	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRS	RESI	DLEC [2:0]			BO	EW	EP	ACT [1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	FDCAN_IR	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x006-0x007F	Reserved	Reserved																															
0x0080	FDCAN_GFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS [1:0]		ANFE [1:0]		RRFS	RRFE
	Reset value																											0	0	0	0	0	0
0x0084	FDCAN_SIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]								FLSSA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0088	FDCAN_XIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]								FLESA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 763. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0090	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0094	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	FIDX[6:0]							MISI [1:0]		BIDX[5:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	FDCAN_NDAT1	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	FDCAN_NDAT2	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	FDCAN_RXF0C	F0OM	F0WM[6:0]							Res.	F0S[6:0]							F0SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00A4	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0	F0F	Res.	Res.	F0PI[5:0]						Res.	Res.	F0GI[5:0]						Res.	RF0FL[6:0]						
	Reset value							0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00A8	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00AC	FDCAN_RXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBSA[13:0]														Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B0	FDCAN_RXF1C	F1OM	F1WM[6:0]							Res.	F1S[6:0]							F1SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B4	FDCAN_RXF1S	DMS [1:0]		Res.	Res.	Res.	Res.	RF1	F1F	Res.	Res.	F1PI[5:0]						Res.	Res.	F1GI[5:0]						Res.	RF1FL[6:0]						
	Reset value	0	0					0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00B8	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00BC	FDCAN_RXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBDS [2:0]				Res.	F1DS [2:0]			Res.	F0DS [2:0]			
	Reset value																				0	0	0			0	0	0			0	0	0
0x00C0	FDCAN_TXBC	Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]						TBSA[13:0]														Res.	Res.
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]					Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
	Reset value											0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00C8	FDCAN_TXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS [2:0]		
	Reset value																														0	0	0

Table 763. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00CC	FDCAN _TXBRP	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D0	FDCAN _TXBAR	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D4	FDCAN _TXBCR	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	FDCAN _TXBTO	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	FDCAN _TXBCF	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E0	FDCAN _TXBTIE	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E4	FDCAN _TXBCIE	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F0	FDCAN _TXEFC	Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]						EFSA[15:2]														Res.	Res.
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00F4	FDCAN _TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEF	EFF	Res.	Res.	Res.	EFPI[4:0]					Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
	Reset value							0	0				0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00F8	FDCAN _TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
	Reset value																												0	0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

72.6 TTCAN registers

These registers are available only for FDCAN1.

72.6.1 FDCAN TT trigger memory configuration register (FDCAN_TTTMC)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TME[6:0]**: Trigger memory elements
0: No trigger memory
1-64: Number of trigger memory elements
>64: Values greater than 64 are interpreted as 64
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TMSA[13:0]**: Trigger memory start address.
Start address of trigger memory in message RAM (32-bit word address, see *Figure 904*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

72.6.2 FDCAN TT reference message configuration register (FDCAN_TTRMC)

Address offset: 0x0104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMPS	XTD	Res.	RID[28:16]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RMPS**: Reference message payload select
Ignored in case of time slaves.
0: Reference message has no additional payload
1: The following elements are taken from Tx buffer 0:
Message marker MM,
Event FIFO control EFC,
Data length code DLC,
Data Bytes DB (level 1: bytes 2-8, levels 0, 2: bytes 5-8)
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 30 **XTD**: Extended identifier
0: 11-bit standard identifier
1: 29-bit extended identifier
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:0 **RID[28:0]**: Reference identifier.
Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.6.3 FDCAN TT operation configuration register (FDCAN_TTOCF)

Address offset: 0x0108

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECS	IRTO[6:0]							LDSDL[2:0]			TM	GEN	Res.	OM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **EVTP**: Event trigger polarity.
0: Rising edge trigger
1: Falling edge trigger
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 25 **ECC**: Enable clock calibration.
0: Automatic clock calibration in FDCAN levels 0, 2 is disabled
1: Automatic clock calibration in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 24 **EGTF**: Enable global time filtering.
0: Global time filtering in FDCAN levels 0, 2 is disabled
1: Global time filtering in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:16 **AWL[7:0]**: Application watchdog limit.
The application watchdog can be disabled by programming AWL to 0x00.
0x00 to FF: Maximum time after which the application must serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 15 **EECS**: Enable external clock synchronization
If enabled, TUR configuration (FDCAN_TURCF.NCL only) may be updated during FDCAN operation.
0: External clock synchronization in FDCAN levels 0, 2 disabled
1: External clock synchronization in FDCAN levels 0, 2 enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 14:8 **IRTO[6:0]**: Initial reference trigger offset.
0x00 to 7F Positive offset, range from 0 to 127
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:5 **LDSDL[2:0]**: LD of synchronization deviation limit.
The synchronization deviation limit SDL is configured by its dual logarithm LDSDL with SDL = 2 * (LDSDL + 5). SDL is comprised between 32 and 4096. It should not exceed the clock tolerance given by the CAN bit timing configuration.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 4 **TM**: Time master.
0: Time master function disabled
1: Potential time master
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 3 **GEN**: Gap enable.
0: Strictly time-triggered operation
1: External event-synchronized time-triggered operation
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **OM[1:0]**: Operation mode.
00: Event-driven CAN communication, default
01: TTCAN level 1
10: TTCAN level 2
11: TTCAN level 0
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

72.6.4 FDCAN TT matrix limits register (FDCAN_TTMLM)

Address offset: 0x010C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ENTT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXEW[3:0]				CSS[1:0]		CCM[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ENTT[11:0]**: Expected number of Tx triggers

0x000 to FFF Expected number of Tx triggers in one matrix cycle.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TXEW[3:0]**: Tx enable window

0x0 to F Length of Tx enable window, 1-16 NTU cycles

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:6 **CSS[1:0]**: Cycle start synchronization

Enables sync pulse output.

00: No sync pulse

01: Sync pulse at start of basic cycle

10: Sync pulse at start of matrix cycle

11: Reserved

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 5:0 **CCM[5:0]:** Cycle count Max

0x00: 1 basic cycle per matrix cycle

0x01: 2 basic cycles per matrix cycle

0x03: 4 basic cycles per matrix cycle

0x07: 8 basic cycles per matrix cycle

0x0F: 16 basic cycles per matrix cycle

0x1F: 32 basic cycles per matrix cycle

0x3F: 64 basic cycles per matrix cycle

Others: Reserved

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: ISO 11898-4, Section 5.2.1 requires that only the listed cycle count values are configured. Other values are possible, but may lead to inconsistent matrix cycles.

72.6.5 FDCAN TUR configuration register (FDCAN_TURCF)

The length of the NTU is given by: NTU = CAN clock period x NC/DC.

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC extends from 0x10000 to 0x1FFFF. The value configured by NCL is the initial value for FDCAN_TURNA.NAV[15:0]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

- Level 1: NC 4 × DC and NTU = CAN bit time
- Levels 0 and 2: NC 8 × DC

The actual value of FDCAN_TUR may be changed by the clock drift compensation function of TTCAN level 0 and level 2 in order to adjust the node local view of the NTU to the time master view of the NTU. DC is not changed by the automatic drift compensation, FDCAN_TURNA.NAV may be adjusted around NC in the range of the synchronization deviation limit given by FDCAN_TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in achieve the best computational accuracy for the drift compensation process.

Address offset: 0x0110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELT	Res.	DC[13:0]													
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NCL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ELT**: Enable local time.

0: Local time is stopped, default

1: Local time is enabled

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: The local time is started by setting ELT, it remains active until ELT is reset or until the next hardware reset. FDCAN_TURCF.DC is locked when FDCAN_TURCF.ELT = 1. If ELT is written to 0, the readable value stays at 1 until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register is locked.

Bit 30 Reserved, must be kept at reset value.

Bits 29:16 **DC[13:0]**: Denominator configuration.

0x0000: Illegal value

0x0001 to 0x3FFF: Denominator configuration

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:0 **NCL[15:0]**: Numerator configuration low.
Write access to the TUR numerator configuration low is only possible during configuration with FDCAN_TURCF.ELT = 0 or if FDCAN_TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT configuration mode, the new value takes effect when FDCAN_TTOST.WECS is cleared to 0. NCL is locked FDCAN_TTOST.WECS is 1.
0x0000 to FFFF Numerator configuration low
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: If NC < 7 × DC in TTCAN level 1, subsequent time marks in the trigger memory must differ by at least two NTUs.

72.6.6 FDCAN TT operation control register (FDCAN_TTOCN)

Address offset: 0x0114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **LCKC**: TT operation control register locked.
Set by a write access to register FDCAN_TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.
0: Write access to FDCAN_TTOCN enabled
1: Write access to FDCAN_TTOCN locked

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ESCN**: External synchronization control
If enabled the FDCAN synchronizes its cycle time phase to an external event signaled by a rising edge at event trigger pin (see *Section 72.4.18: Synchronization to external time schedule*).
0: External synchronization disabled
1: External synchronization enabled

Bit 12 **NIG**: Next is gap.
This bit can be set only when the FDCAN is the actual time master and when it is configured for external event-synchronized time-triggered operation (FDCAN_TTOCF.GEN = 1)
0: No action, reset by reception of any reference message
1: Transmit next reference message with Next_is_Gap = 1

Bit 11 **TMG**: Time mark gap.
0: Reset by each reference message
1: Next reference message started when register time mark interrupt FDCAN_TTIR.RTMI is activated

Bit 10 **FGP**: Finish gap.
Set by the CPU, reset by each reference message
0: No reference message requested
1: Application requested start of reference message

Bit 9 **GCS**: Gap control select
0: Gap control independent from event trigger
1: Gap control by input event trigger pin

Bit 8 **TTIE**: Trigger time mark interrupt pulse enable
External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Schedule or In_Gap.
0: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance disabled
1: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance enabled

Bits 7:6 **TMC[1:0]**: Register time mark compare.
00: No register time mark interrupt generated
01: Register time mark interrupt if time mark = cycle time
10: Register time mark interrupt if time mark = local time
11: Register time mark interrupt if time mark = global time

Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = 00, then reconfigure FDCAN_TTTMK, and finally set FDCAN_TMC to the intended time reference.

Bit 5 **RTIE**: Register time mark interrupt pulse enable.
Register time mark interrupts are configured by register FDCAN_TTTMK. A register time mark interrupt pulse with the length of one fdcan_tq_ck period is generated when time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Register time mark interrupt output disabled
1: Register time mark interrupt output enabled

Bits 4:3 **SWS[1:0]**: Stop watch source.
00: Stop watch disabled
01: Actual value of cycle time is copied to FDCAN_TTCPT.SWV
10: Actual value of local time is copied to FDCAN_TTCPT.SWV
11: Actual value of global time is copied to FDCAN_TTCPT.SWV

Bit 2 **SWP**: Stop watch polarity.
0: Rising edge trigger
1: Falling edge trigger

Bit 1 **ECS**: External clock synchronization.
Writing a 1 to ECS sets FDCAN_TTOST.WECS if the node is the actual time master. ECS is reset after one APB clock period. The external clock synchronization takes effect at the start of the next basic cycle.

Bit 0 **SGT**: Set global time.
Writing a 1 to SGT sets FDCAN_TTOST.WGDT if the node is the actual time master. SGT is reset after one APB clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to FDCAN_TTGTP.

72.6.7 FDCAN TT global time preset register (FDCAN_TTGTP)

If TTOST.WGDT is set, the next reference message is transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

Address offset: 0x0118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CTP[15:0]**: Cycle time target phase
CTP is write-protected while FDCAN_TTOCN.ESCN or FDCAN_TTOST.SPL are set (see *Section 72.4.18: Synchronization to external time schedule*).
0x0000 to FFFF defines the target value of cycle time when a rising edge of event trigger is expected

Bits 15:0 **TP[15:0]**: Time preset
TP is write-protected while FDCAN_TTOST.WGTD is set.
0x0000 to 7FFF next master reference mark = master reference mark + TP
0x8000 reserved
0x8001–FFFF Next master reference mark = master reference mark - (0x10000 - TP).

72.6.8 FDCAN TT time mark register (FDCAN_TTTMK)

A time mark interrupt (FDCAN_TTIR.TMI = 1) is generated when the time base indicated by FDCAN_TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

Address offset: 0x011C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]						
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LCKM**: TT time mark register locked

Always set by a write access to registers FDCAN_TTOCN. Set by write access to register FDCAN_TTTMK when FDCAN_TTOCN.TMC 00. Reset when the registers have been synchronized into the CAN clock domain.

0: Write access to FDCAN_TTTMK enabled

1: Write access to FDCAN_TTTMK locked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **TICC[6:0]**: Time mark cycle code

Cycle count for which the time mark is valid.

0b000000x valid for all cycles

0b000001c valid every second cycle at cycle count mod2 = c

0b00001cc valid every fourth cycle at cycle count mod4 = cc

0b0001ccc valid every eighth cycle at cycle count mod8 = ccc

0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc

0b01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc

0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc

Bits 15:0 **TM[15:0]**: Time mark

0x0000 to FFFF time mark

Note: When using byte access to register FDCAN_TTTMK it is recommended to first disable the time mark compare function (FDCAN_TTOCN.TMC = 00) to avoid comparisons on inconsistent register values.

72.6.9 FDCAN TT interrupt register (FDCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

Address offset: 0x0120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CER**: Configuration error

Trigger out of order.

0: No error found in trigger list

1: Error found in trigger list

Bit 17 **AW**: Application watchdog

0: Application watchdog served in time

1: Application watchdog not served in time

Bit 16 **WT**: Watch trigger
0: No missing reference message
1: Missing reference message (level 0: cycle time 0xFF00)

Bit 15 **IWTG**: Initialization watch trigger
The initialization is restarted by resetting IWT.
0 No missing reference message during system startup
1 No system startup due to missing reference message

Bit 14 **ELC**: Error level changed
Not set when error level changed during initialization.
0: No change in error level
1: Error level changed

Bit 13 **SE2**: Scheduling error 2
0: No scheduling error 2
1: Scheduling error 2 occurred

Bit 12 **SE1**: Scheduling error 1
0: No scheduling error 1
1: Scheduling error 1 occurred

Bit 11 **TXO**: Tx count overflow
0: Number of Tx trigger as expected
1: More Tx trigger than expected in one cycle

Bit 10 **TXU**: Tx count underflow
0: Number of Tx trigger as expected
1: Less Tx trigger than expected in one cycle

Bit 9 **GTE**: Global time error
Synchronization deviation SD exceeds limit specified by FDCAN_TTOCF.LDSDL, TTCAN levels 0, 2 only.
0: Synchronization deviation within limit
1: Synchronization deviation exceeded limit

Bit 8 **GTD**: Global time discontinuity
0: No discontinuity of global time
1: Discontinuity of global time

Bit 7 **GTW**: Global time wrap
0: No global time wrap occurred
1: Global time wrap from 0xFFFF to 0x0000 occurred

Bit 6 **SWE**: Stop watch event
0: No rising/falling edge at stop watch trigger pin detected
1 :Rising/falling edge at stop watch trigger pin detected

Bit 5 **TTMI:** Trigger time mark event internal
Internal time mark events are configured by trigger memory element TMIN (see *Section 72.4.24*). Set when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Gap or In_Schedule.
0: Time mark not reached
1: Time mark reached (level 0: cycle time FDCAN_TTOCF.RTO x 0x200)

Bit 4 **RTMI**: Register time mark interrupt
Set when time referenced by TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Time mark not reached
1: Time mark reached

Bit 3 **SOG**: Start of gap
0 No reference message seen with Next_is_Gap bit set
1 Reference message with Next_is_Gap bit set becomes valid

Bit 2 **CSM**: Change of synchronization mode
0: No change in master to slave relation or schedule synchronization
1: Master to slave relation or schedule synchronization changed,
also set when FDCAN_TTOST.SPL is reset

Bit 1 **SMC**: Start of matrix cycle
0: No matrix cycle started since bit has been reset
1: Matrix cycle started

Bit 0 **SBC**: Start of basic cycle
0: No basic cycle started since bit has been reset
1: Basic cycle started

72.6.10 FDCAN TT interrupt enable register (FDCAN_TTIE)

The settings in the TT interrupt enable register determine which status changes in the TT interrupt register result in an interrupt.

Address offset: 0x0124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERE**: Configuration error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 17 **AWE**: Application watchdog interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 16 **WTE**: Watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 15 **IWTE**: Initialization watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 14 **ELCE**: Change error level interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 13 **SE2E**: Scheduling error 2 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 12 **SE1E**: Scheduling error 1 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 11 **TXOE**: Tx count overflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 10 **TXUE**: Tx count underflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 9 **GTEE**: Global time error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 8 **GTDE**: Global time discontinuity interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 7 **GTWE**: Global time wrap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 6 **SWEE**: Stop watch event interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 5 **TTMIE:** Trigger time mark event internal interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 4 **RTMIE**: Register time mark interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 3 **SOGE**: Start of gap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 2 **CSME**: Change of synchronization mode interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 1 **SMCE**: Start of matrix cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 0 **SBCE**: Start of basic cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

72.6.11 FDCAN TT interrupt line select register (FDCAN_TTILS)

The TT interrupt line select register assigns an interrupt generated by a specific interrupt flag from the TT interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0128

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERL**: Configuration error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 17 **AWL**: Application watchdog interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 16 **WTL:** Watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 15 **IWTL:** Initialization watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 14 **ELCL:** Change error level interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 13 **SE2L:** Scheduling error 2 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 12 **SE1L:** Scheduling error 1 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 11 **TXOL:** Tx count overflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 10 **TXUL:** Tx count underflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 9 **GTEL:** Global time error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 8 **GTDL:** Global time discontinuity interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 7 **GTWL:** Global time wrap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 6 **SWEL:** Stop watch event interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 5 TTMIL: Trigger time mark event internal interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 4 **RTMIL:** Register time mark interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 3 **SOGL:** Start of gap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 2 **CSML:** Change of synchronization mode interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 1 **SMCL:** Start of matrix cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 0 **SBCL:** Start of basic cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

72.6.12 FDCAN TT operation status register (FDCAN_TTOST)

Address offset: 0x012C

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **SPL**: Schedule phase lock

The bit is valid only when external synchronization is enabled (FDCAN_TTOCN.ESCN = 1). In this case it signals that the difference between cycle time configured by FDCAN_TTGTP.CTP and the cycle time at the rising edge at event trigger pin is less than or equal to 9 NTU (see *Section 72.4.18: Synchronization to external time schedule*).

0: Phase outside range

1: Phase inside range

Bit 30 **WECS**: Wait for external clock synchronization.

0: No external clock synchronization pending

1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.

Bit 29 **AWE**: Application watchdog event

The application watchdog is served by reading FDCAN_TTOST. When the watchdog is not served in time, bit AWE is set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode.

0: Application watchdog served in time

1: Failed to serve application watchdog in time

Bit 28 **WFE**: Wait for event

0: No gap announced, reset by a reference message with Next_is_Gap = 0

1: Reference message with Next_is_Gap = 1 received

Bit 27 **GSI**: Gap started indicator

0: No gap in schedule, reset by each reference message and for all time slaves

1: Gap time after basic cycle has started

Bits 26:24 **TMP[2:0]**: Time master priority

0x0-7 Priority of actual time master

Bit 23 **GFI**: Gap finished indicator

Set when the CPU writes FDCAN_TTOCN.FGP, or by a time mark interrupt if TMG = 1, or via input pin (event trigger) if FDCAN_TTOCN.GCS = 1. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.

0: Reset at the end of each reference message

1: Gap finished by FDCAN

Bit 22 **WGTD**: Wait for global time discontinuity

0: No global time preset pending

1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = 1 or after it received a reference message.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:8 **RTO[7:0]**: Reference trigger offset

The reference trigger offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the FDCAN becomes time master (MS[1:0] = 11), the reset of RTO is delayed due to synchronization between user and CAN clock domain. For time slaves the value configured by FDCAN_TTOCF.IRTO is read.

0x00-FF Actual reference trigger offset value

Bit 7 **QCS**: Quality of clock speed

Only relevant in TTCAN level 0 and level 2, otherwise fixed to 1.

0: Local clock speed not synchronized to time master clock speed

1: Synchronization deviation ≤ SDL

Bit 6 **QGTP**: Quality of global time phase
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 0.
0: Global time not valid
1: Global time in phase with time master

Bits 5:4 **SYS[1:0]**: Synchronization state
00: Out of Synchronization
01: Synchronizing to FDCAN communication
10: Schedule suspended by gap (In_Gap)
11: Synchronized to schedule (In_Schedule)

Bits 3:2 **MS[1:0]**: Master state
00: Master_Off, no master properties relevant
01: Operating as time Slave
10: Operating as backup time master
11: Operating as current time master

Bits 1:0 **EL[1:0]**: Error level
00: Severity 0 - No error
01: Severity 1 - Warning
10: Severity 2 - error
11: Severity 3 - Severe error

72.6.13 FDCAN TUR numerator actual register (FDCAN_TURNA)

There is no drift compensation in TTCAN level 1 (NAV = NC). In TTCAN level 0 and level 2, the drift between the node local clock and the time master local clock is calculated. The drift is compensated when the synchronization deviation (difference between NC and the calculated NAV) is lower than 2 * (FDCAN_TTOCF.LDSDL + 5). With FDCAN_TTOCF.LDSDL < 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

Address offset: 0x0130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:16]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NAV[17:0]**: Numerator actual value
0x0EFFF: illegal value
0x0F000 to 20FFF: actual numerator value
0x21000: illegal value

72.6.14 FDCAN TT local and global time register (FDCAN_TTLGT)

Address offset: 0x0134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **GT[15:0]**: Global time
Non-fractional part of the sum of the node local time and its local offset (see *Section 72.4.13*).
0x0000 to FFFF Global time value of FDCAN network

Bits 15:0 **LT[15:0]**: Local time
Non-fractional part of local time, incremented once each local NTU (see *Section 72.4.13*).
0x0000 to FFFF Local time value of FDCAN node

72.6.15 FDCAN TT cycle time and count register (FDCAN_TTCTC)

Address offset: 0x0138

Reset value: 0x003F 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CC[5:0]**: Cycle count
0x00 to 3F Number of actual basic cycle in the system matrix

Bits 15:0 **CT[15:0]**: Cycle time
Non-fractional part of the difference of the node local time and Ref_Mark (see *Section 72.4.13*).
0x0000 to FFFF: cycle time value of FDCAN basic cycle

72.6.16 FDCAN TT capture time register (FDCAN_TTCPT)

Address offset: 0x013C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
										r	r	r	r	r	r

Bits 31:16 **SWV[15:0]**: Stop watch value

On a rising / falling edge (as configured via FDCAN_TTOCN.SWP) at the stop watch trigger pin, when FDCAN_TTOCN.SWS] is different from 00 and FDCAN_TTIR.SWE is 0, the actual time value as selected by FDCAN_TTOCN.SWS (cycle, local, global) is copied to SWV and TFDCAN_TIR.SWE is set to 1.Capturing of the next stop watch value is enabled by resetting FDCAN_TTIR.SWE.

0x0000 to FFFF Captured stop watch value

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CCV[5:0]**: Cycle count value

Cycle count value captured together with SWV.

0x00 to 3F: captured cycle count value

72.6.17 FDCAN TT cycle sync mark register (FDCAN_TTCSM)

Address offset: 0x0140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSM[15:0]**: Cycle sync mark

The cycle sync mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.

0x0000 to FFFF Captured cycle time

72.6.18 FDCAN TT trigger select register (FDCAN_TTTS)

The settings in the FDCAN_TTTS register select the input to be used as event trigger and stop watch trigger.

Address offset: 0x0300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTDEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **EVTSEL[1:0]**: Event trigger input selection
These bits are used to select the input to be used as event trigger
00: fdcan1_evt0
01: fdcan1_evt1
10: fdcan1_evt2
11: fdcan1_evt3

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SWTDEL[1:0]**: Stop watch trigger input selection
These bits are used to select the input to be used as stop watch trigger
00: fdcan1_swt0
01: fdcan1_swt1
10: fdcan1_swt2
11: fdcan1_swt3

72.6.19 FDCAN TT register map

Table 764. FDCAN TT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0100	FDCAN _TTTMC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]							TMSA[13:0]														Res.	Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0104	FDCAN _TTRMC	RMPS	XTD	Res.	RID[28:0]																												
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0108	FDCAN _TTOCF	Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]								EECS	IRTO[6:0]							LDSDL [2:0]			TM	GEN	Res.	OM [1:0]	
	Reset value						0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x010C	FDCAN _TTMLM	Res.	Res.	Res.	Res.	ENTT[11:0]												Res.	Res.	Res.	Res.	TXEW[3:0]				CSS [1:0]		CCM[5:0]					
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 764. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0110	FDCAN_TURCF	ELT	Res.	DC[13:0]														NCL[15:0]															
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114	FDCAN_TTOCN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0118	FDCAN_TTGTP	CTP[15:0]																TP[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x011C	FDCAN_TTTMK	LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]							TM[15:0]															
	Reset value	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	FDCAN_TTIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT	IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124	FDCAN_TTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0128	FDCAN_TTILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x012C	FDCAN_TTOST	SPL	WECS	AWE	WFE	GSI	TMP [2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.	RTO[7:0]								QCS	QGTP	SYS [1:0]		MS [1:0]		EL [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0130	FDCAN_TURNA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	FDCAN_TTLGT	GT[15:0]																LT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0138	FDCAN_TTCTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]						CT[15:0]															
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x013C	FDCAN_TTCPT	SWV[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x0140	FDCAN_TTCSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0144 to 0x01FC	Reserved																																
	Reset value																																

Table 764. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0300	FDCAN_TTTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTSEL[1:0]	
	Reset value																											0	0			0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

72.7 CCU registers

72.7.1 Clock calibration unit core release register (FDCAN_CCU_CREL)

Address offset: 0x0000

Reset value: 0x1114 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 1

Bits 27:24 **STEP[3:0]**: Step of core release = 1

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day = 18

72.7.2 Calibration configuration register (FDCAN_CCU_CCFG)

Address offset: 0x0004

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]			
rw												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWR**: Software reset

Writing a 1 to this bit resets the calibration FSM to state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). The calibration watchdog value CWD.WDV is also reset. Registers FDCAN_CCFG, FDCAN_CCU_CSTAT and the calibration watchdog configuration CWD.WDC are unchanged. The bit remains set until reset is completed.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:16 **CDIV[3:0]**: Clock divider

The clock divider must be configured when the clock calibration is bypassed (BCC = 1) to ensure that the FDCAN requirement is fulfilled.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bit FDCAN_CCCR.CCE = 1.

Bits 15:8 **OCPM[7:0]**: Oscillator clock periods minimum

Configures the minimum number of periods in two CAN bit times. OCPM is used in basic calibration to avoid false measurements in case of glitches on the bus line. The configured number of periods is OCPM × 32. The configuration depends on the frequency and the bitrate configured in FDCAN modules (from 125 kbit/s up to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz fdcan_ker_ck and 1 Mbit/s CAN bitrate.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 7 **CFL**: Calibration field length

0: Calibration field length is 32 bits
1: Calibration field length is 64 bits

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 6 **BCC**: Bypass clock calibration

If this bit is set, the clock input fdcan_ker_ck is routed to the time quanta clock through a clock divider configurable via CDIV. In this case the baudrate prescaler of the connected FDCANs must be configured to generate the FDCAN internal time quanta clock.

0: Clock calibration unit generates time quanta clock
1: Clock calibration unit bypassed (default configuration)

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **TQBT[4:0]**: Time quanta per bit time

Configures the number of time quanta per bit time. Same value as configured in FDCAN modules. The range of the resulting time quanta clock fdcan_tq_ck is from 0.5 MHz (bitrate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bitrate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.
Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

72.7.3 Calibration status register (FDCAN_CCU_CSTAT)

Address offset: 0x0008

Reset value: 0x0203 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALS[1:0]		Res.	TQC[10:0]											OCPC[17:16]	
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 **CALS[1:0]**: Calibration state

00: Not_Calibrated
01: Basic_Calibrated
10: Precision_Calibrated
11: Reserved

Bit 29 Reserved, must be kept at reset value.

Bits 28:18 **TQC[10:0]**: Time quanta counter

Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

Bits 17:0 **OCPC[17:0]**: Oscillator clock period counter

Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

72.7.4 Calibration watchdog register (FDCAN_CCU_CWD)

Address offset: 0x000C

Reset value: 0x0000 0000

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). In this state the calibration watchdog monitors the message received. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM stays in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01), the calibration watchdog is restarted with each received message . In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM returns to state

Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message is received, state Precision_Calibrated (FDCAN_CCU_CSTAT.CALS = 10) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input. In case no message from a quartz controlled node is received by the attached TTCAN until the calibration watchdog has counted down to 0, the calibration FSM transits back to state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01). The signal is active when the CAN protocol engine on the attached TTCAN is started i.e. when the INIT bit is reset.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE, the interrupt line is activated (set to high). Interrupt line remains active until the interrupt flag FDCAN_CCU_IR.CWE is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WDV[15:0]**: Watchdog value

Actual calibration watchdog counter value.

Bits 15:0 **WDC[15:0]**: WDC

Watchdog configuration

Start value of the calibration watchdog counter. With the reset value of 00 the counter is disabled. Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

72.7.5 Clock calibration unit interrupt register (FDCAN_CCU_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register. The configuration of FDCAN_CCU_IE controls whether an interrupt is generated or not.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSC**: Calibration state changed
0: Calibration state unchanged
1: Calibration state has changed

Bit 0 **CWE**: Calibration watchdog event
0: No calibration watchdog event
1: Calibration watchdog event occurred

72.7.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE)

Address offset: 0x0014

Reset value: 0x0000 0000

The settings in the CU interrupt enable register determine whether a status change in the CU interrupt register is signaled on an interrupt line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSCE**: Calibration state changed enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **CWEE**: Calibration watchdog event enable
0: Interrupt disabled
1: Interrupt enabled

72.7.7 CCU register map

Table 765. CCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CCU_CREL	REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_CCU_CCFG	SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]				OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
	Reset value	0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	FDCAN_CCU_CSTAT	CALS[1:0]		Res.	TQC[10:0]											OCPC[17:0]																	
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	FDCAN_CCU_CWD	WDV[15:0]																WDC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 765. CCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010	FDCAN_ CCU_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
	Reset value																															0	0
0x0014	FDCAN_ CCU_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
	Reset value																															0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

73 USB subsystem (USBSS)

73.1 Overview

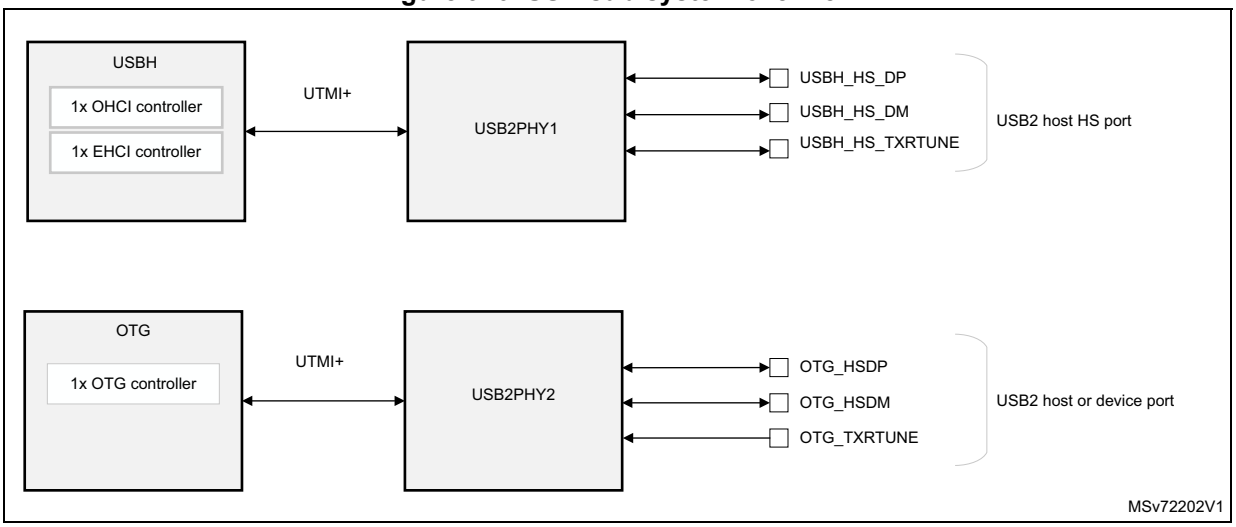
The USB sub-system includes one USB2 host high-speed port and one USB2.0 host/device high-speed port.

The USB2 host high-speed port also supports full speed and low speed modes.

The USB2.0 host/device high-speed port can be configured statically as host or device. It is called OTG for legacy reason. In addition to high-speed mode, it also supports full speed and low speed modes when acting as host, and full speed mode when acting as device.

A high level block diagram is shown below.

Figure 916. USB sub-system overview



73.2 USB2 host high-speed port

73.2.1 USB2 host high-speed port main features

- Supports OHCI model for full speed and low speed operation
- Supports EHCI model for high-speed operation
- Supports LPM (link power management)
- USB2PHY1 supports high-speed, full speed and low speed modes
- DMA engine with local 512 Bytes packet buffer
- Supports battery charging v1.2, with the exception of the accessory charger adapter mode
- Support suspend, sleep, resume, and remote wakeup operations
- Provides electrical parameter override bits for optimal interoperability

The supported standards are:

- Universal serial bus specification, revision 2.0, USB implementers forum, Inc., April 27, 2000
- Enhanced host controller interface specification for universal serial bus, revision 1.0, March 12, 2002
- EHCI 1.1 addendum, version 0.6, October 2007
- Open host controller interface specification for USB, release 1.0a, September 14, 1999
- Engineering change notice: USB2.0 link power management addendum, July 16, 2007
- UTMI+ specification, revision 1.0, Philips semiconductor, February 25, 2004
- Battery charging specification, revision 1.2, December 7, 2010

73.2.2 USB2 host high-speed port implementation

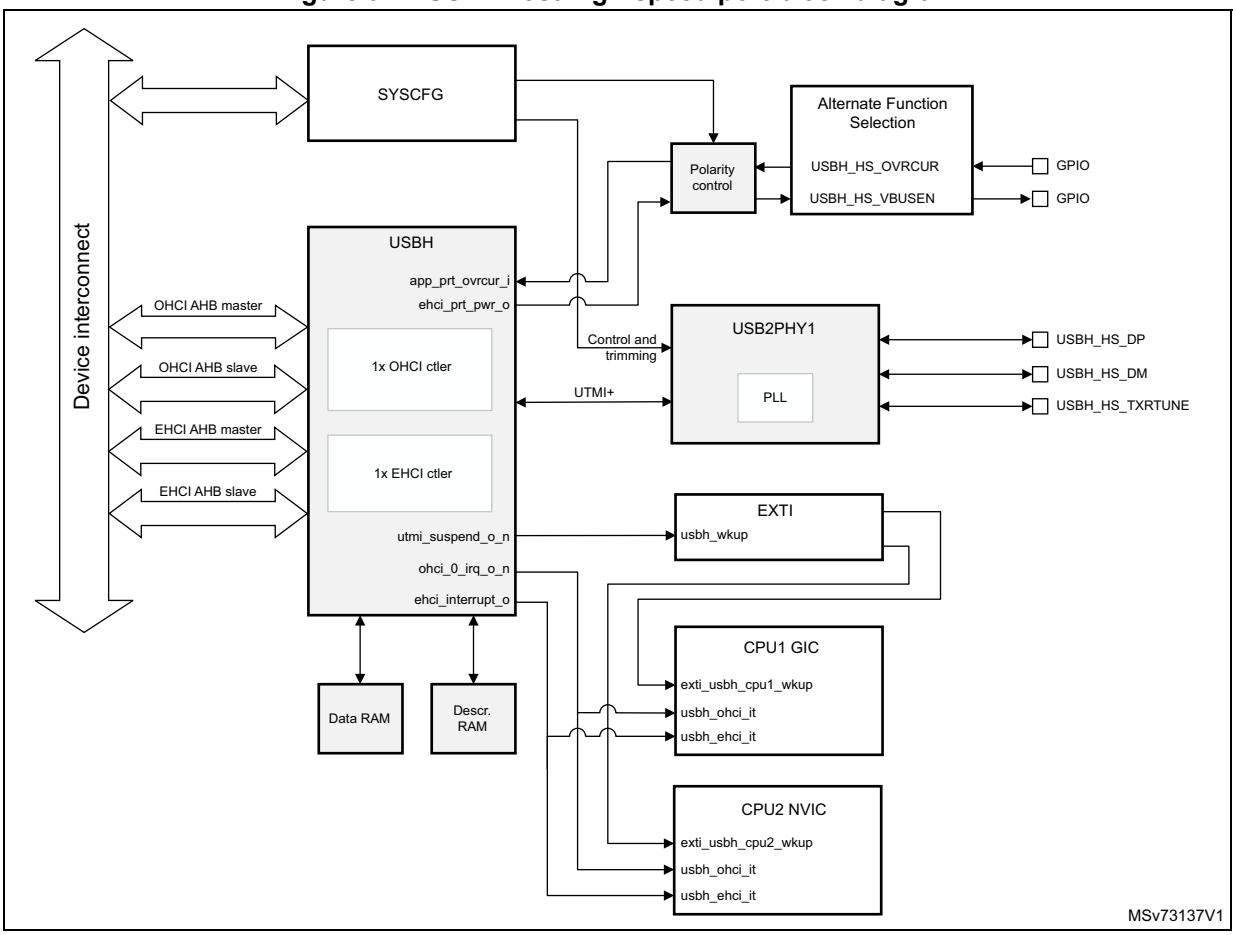
Table 766. Implementation parameters

Parameter	Value
EHCI descriptors/data prefetching	Yes
Number of OHCI controller	1
Number of USB2 physical port	1
Dedicated AHB interfaces for EHCI controller and OHCI controller	Yes
EHCI controller includes port power control	Yes
EHCI packet buffer depth: 512 Bytes	512 Bytes
Programmable packet buffer depth during runtime	Yes
EHCI extended capabilities pointer	0xA0
Supports EHCI asynchronous schedule park capability	Yes
Supports EHCI programmable frame list flag	Yes
64-bit addressing capability for EHCI	No
Supports HSIC functionality	No
Supports heterogeneous interface per port	No, only UTMI+ is supported
ULPI interface	No
UTMI legacy interface	No, UTMI+ is selected
OHCI PowerOn to PowerGood time	2 ms
Is the device connected to port #1 non removable	No
Support link power management for EHCI controller	Yes
Support OHCI legacy	No
Enhanced AHB master INCRX support	Yes
Support power switching for OHCI-owned port	Yes
Global or per port power switching for OHCI-owned port	Per port
Port power control mask for port #1	0

73.2.3 USB2 host high-speed port functional description

USB2 host high-speed port block diagram

Figure 917. USB2 host high-speed port block diagram



The AHB slave ports of USBH are protected by two RISUP instances and the traffic from the AHB master ports of USBH is filtered by two RIMU instances. Both RISUP and RIMU instances are located at device top level.

USB2 host high-speed port pins and internal signals

Table 767. USB2 host high-speed port pins

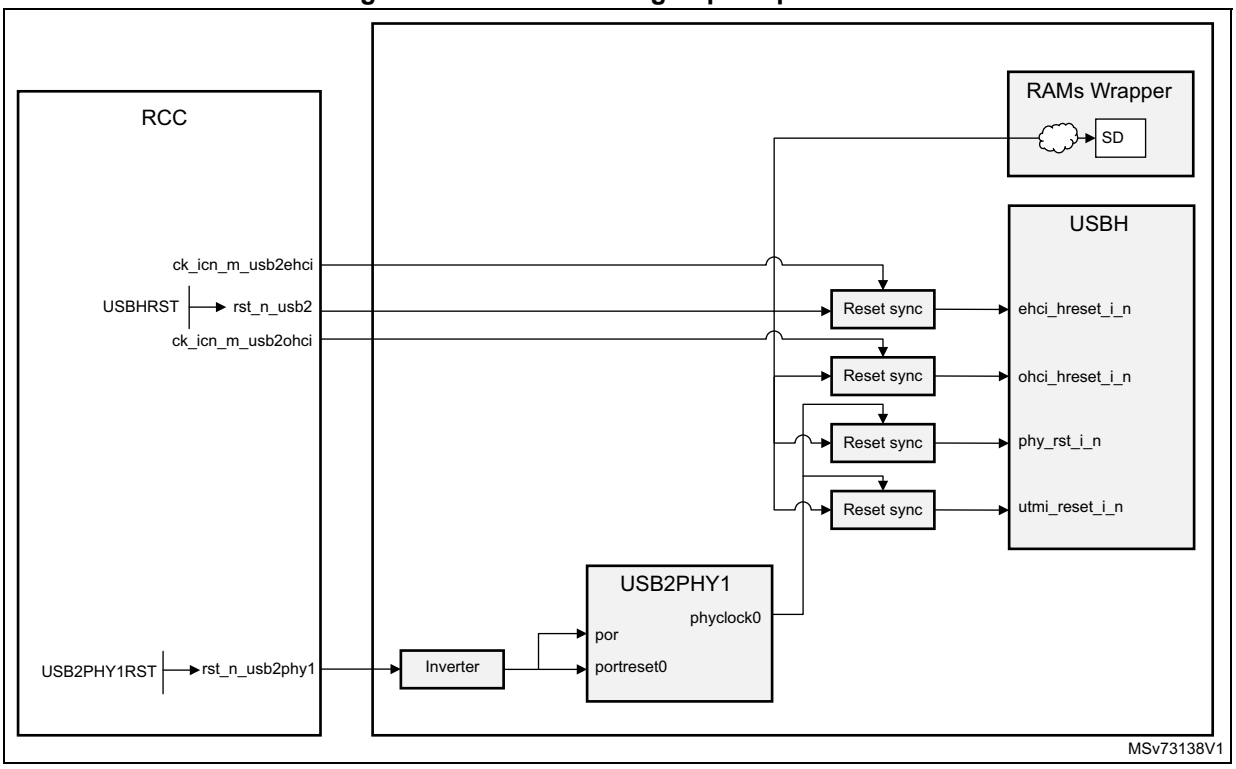
Signal name	Signal type	Description
USBH_HS_DP	3.3 V analog	USB D+ signal
USBH_HS_DM	3.3 V analog	USB D- signal
USBH_HS_TXRTUNE	1.8 V analog	Transmitter resistor tune pin This analog signal connects to the ground through an external 200 Ω ±1% resistor that adjusts the high-speed source impedance of USB2PHY1.
USBH_HS_OVRCUR	V_{DD} domain	Alternate input function of a GPIO The application can use this input to inform the USB2 host controller of an over-current situation. When asserted, the USB2 host high-speed port enters disable state. The polarity of this input is defined by the USBH_OVRCUR_POLARITY bit in SYSCFG_USBHCR register.
USBH_HS_VBUSEN	V_{DD} domain	Alternate output function of a GPIO The application can use this output to switch on or off the V_{BUS} source of the USB2 host high-speed port. The polarity of this output is defined by the USBH_VBUSEN_POLARITY bit in SYSCFG_USBHCR register.

Table 768. USB2 host high-speed port internal signals

Signal name	Signal type	Description
OHCI AHB master	Digital	AHB master interface to the peripheral interconnects for memory access of OHCI engine. AHBLite protocol, 32-bit data width.
OHCI AHB slave	Digital	AHB slave interface for access to OHCI registers. AHBLite protocol, 32-bit data width.
EHCI AHB master	Digital	AHB master interface to the peripheral interconnects for memory access of EHCI engine. AHBLite protocol, 32-bit data width.
EHCI AHB slave	Digital	AHB slave interface for access to EHCI registers. AHBLite protocol, 32-bit data width.

USB2 host high-speed port resets

Figure 918. USB2 host high-speed port reset



There are two resets for the USB2 host high-speed port:

- rst_n_usb2phy1 which is controlled by the USB2PHY1RST bit in RCC_USB2PHY1CFGR register.
- rst_n_usb2 which is controlled by the USBHRST bit in RCC_USBHCFGR register.

When configuring the USB2 host high-speed host, the reset sequence to be applied is:

1. both rst_n_usb2phy1 and rst_n_usb2 are asserted2
2. the frequency of USB2PHY1 reference clock is selected in SYSCFG_USB2PHY1CR register (it is assumed that the SYSCFG block has been taken out of reset previously)
3. USB2PHY1 trimming bits are programmed (if needed)
4. clocks coming from RCC or oscillator are enabled
5. rst_n_usb2phy1 is released
6. wait for 50 µs (USB2PHY1 PLL lock time and start of generation of phyclock0)
7. release rst_n_usb2

USB2 host high-speed port clocking

Refer to the USB clocking schematic in the RCC user specifications.

USB2 host high-speed host memory requirements

Table 769. USB2 host high-speed host memory requirements

Port	Memory area required
OHCI AHB slave	1 KByte
EHCI AHB slave	1 KByte

Description of the USB2 host controller (USBH)

Refer to *Section 75: USB2 EHCI/OHCI high/full/low-speed host controller (USBH)*.

Description of the USB2PHY1

Refer to *Section 23.2: SYSCFG registers*.

73.2.4 USB2 host high-speed host interrupts

The USB2 host high-speed host generates two interrupts.

Table 770. USB2 host high-speed host interrupts

USBH pin name	CPU1 interrupt	CPU2 interrupt
OHCI_0_irq_o_n	usbh_ohci_it	usbh_ohci_it
ehci_interrupt_0	usbh_ehci_it	usbh_ehci_it

The USB2 host high-speed host generates one wake-up event which is turned into an interrupt by the EXTI block. This wake-up event can be used to wake-up the system from Stop mode.

Table 771. USB2 host high-speed host wake-up event

USBH pin name	EXTI event	CPU1 interrupt	CPU2 interrupt
utmi_suspend_o_n	USBH WAKEUP	exti_usbh_cpu1_wkup	exti_usbh_cpu2_wkup

When the USB2 host high-speed host is inactive, utmi_suspend_o_n is asserted and places USB2PHY1 in a minimal power mode and shuts down all blocks not necessary for resume operation. The system can then move to Stop mode if desired. When a remote wake-up event is received, utmi_suspend_o_n is de-asserted asynchronously which triggers the usbh_wkup event and the system wakes up from Stop mode. Exti_usbh_cpu1_wkup (resp. exti_usbh_cpu2_wkup) interrupt is generated by the EXTI block which allows the USB driver to resume operation.

73.2.5 USB2PHY1 integration

In addition to the polarity control of USBH_HS_OVRCUR and USBH_HS_VBUSEN alternate functions, and the selection of USB2PHY1 reference clock frequency, some SYSCFG bits are used to control the battery charging procedure and the trimming of electrical parameters in USB2PHY1.

Battery charging

USB2PHY1 integrates the voltage and current sources, as well as the voltage comparators required to support battery charging v1.2. The relevant control bits are in SYSCFG_USB2PHY1BCCR and SYSCFG_USB2PHY1BCSR registers.

The default value in SYSCFG_USB2PHY1BCCR register are such that the USB2 host high-speed port behaves as a standard downstream port (SDP).

If the application wants the USB2 host high-speed port to behave as a charging downstream port (CDP), one of the following two control sequences shall be implemented:

Control sequence 1:

- Set the H_CDPEN bit to 1.
- Decode the disconnect information transmitted by the USB2PHY1 to the USBH controller (HOSTDISCONNECT0 output of USB2PHY1).
  - If disconnect = 1, set the H_VDMSRCEN bit to 1.
  - Else set the H_VDMSRCEN bit to 0.

Control sequence 2:

- Set the H_CDPEN bit to 1 and the H_CDPDETEN bit to 1.
- Monitor CHGDET and FSVPLUS status bits:
  - If CHGDET = 1 and FSVPLUS = 0, set the H_VDMSRCEN bit to 1
  - Else, set the H_VDMSRCEN bit to 0

Trimming of USB2PHY1 electrical parameters

Some parameters inside the USB2PHY1 PHY can be adjusted by control bits in SYSCFG_USB2PHY1TRIM1CR and SYSCFG_USB2PHY1TRIM2CR registers. Those adjustments must be programmed while the USB2PHY1 is under reset.

Adjustments apply to: the internal PLL, the disconnect threshold, the squelch threshold, the data detection threshold, the transmitter high-speed crossover point, the FS/LS source impedance, the HS DC voltage level, the HS transmitter rise/fall times, the driver source impedance, the HS transmitter pre-emphasis current, the HS transmitter pre-emphasis duration.

The default values in SYSCFG_USB2PHY1TRIM1CR and SYSCFG_USB2PHY1TRIM2CR registers correspond to the recommended values.

73.2.6 Connections to SYSCFG block

Table 772. connections to SYSCFG block

SYSCFG bitfield	HSIF cluster pin name	HS IF dir.	Description
USB2PHY1SEL[2:0]	sysconf_usb2phy1_sel[2:0]	in	Selection of USB2PHY1 reference clock frequency
USB2PHY1CMN	sysconf_usb2phy1_cmn	in	Control of the power down of USB2PHY1 analog blocks during Suspend and Sleep
AUTORSMENB1	sysconf_usb2phy1_autorsmemb_i	in	Enable of USB2PHY1 auto resume mode
RETENABLEN1	sysconf_usb2phy1_retenablen_i	in	Enable of USB2PHY1 retention mode
H_FORCEBCMODE	sysconf_usb2phy1_forcebcmode	in	Forces USB2PHY1 in non-driving, Suspend mode (for battery charging procedure)
H_VDMSRCEN	sysconf_usb2phy1_h_vdatsrcenb0_i	in	Enable of voltage source on USBH_DM for CDP port (for battery charging procedure)
H_CDPDETEN	sysconf_usb2phy1_h_vdatdetenb0_i	in	Enable of voltage detector on USBH_DP for CDP port (for battery charging procedure)
H_CDPEN	sysconf_usb2phy1_chrgsel0_i	in	Enable CDP behavior for USB2 host high-speed port (for battery charging procedure)
FSVMINUS	sysconf_usb2phy1_fsvminus0_o	out	voltage level on USBH_DM (also output of the comparison with VLGC threshold as defined in BC v1.2 standard)
FSVPLUS	sysconf_usb2phy1_fsvplus0_o	out	Voltage level on USBH_DP (also output of the comparison with VLGC threshold as defined in BC v1.2 standard)
CHGDET	sysconf_usb2phy1_chgdet0_o	out	Comparison of voltage on USBH_DP with VDAT_REF threshold as defined in BC v1.2 standard
USB2PHY1TXRESTUNE[1:0]	sysconf_usb2phy1_txrestune[1:0]	in	USB2PHY1 source impedance adjustment
USB2PHY1TXRISETUNE[1:0]	sysconf_usb2phy1_txrisetune[1:0]	in	USB2PHY1 HS transmitter rise/fall time adjustment
USB2PHY1TXVREFTUNE[3:0]	sysconf_usb2phy1_txvreftune[3:0]	in	USB2PHY1 HS DC voltage level adjustment
USB2PHY1TXFSLSTUNE[3:0]	sysconf_usb2phy1_txfslstune[3:0]	in	USB2PHY1 FS/LS source impedance adjustment

Table 772. connections to SYSCFG block (continued)

SYSCFG bitfield	HSIF cluster pin name	HS IF dir.	Description
USB2PHY1TXHSXVTUNE[1:0]	sysconf_usb2phy1_txhsxvtune[1:0]	in	USB2PHY1 transmitter HS crossover adjustment
USB2PHY1OTGTUNE[2:0]	sysconf_usb2phy1_otgtune[2:0]	in	USB2PHY1 VBUS valid threshold adjustment
USB2PHY1VDATREFTUNE[1:0]	sysconf_usb2phy1_vdatreftune[1:0]	in	USB2PHY1 data detect voltage adjustment
USB2PHY1SQRXTUNE[2:0]	sysconf_usb2phy1_sqrxtune[2:0]	in	USB2PHY1 squelch threshold adjustment
USB2PHY1COMPDISTUNE[2:0]	sysconf_usb2phy1_compdistune[2:0]	in	USB2PHY1 disconnect threshold adjustment
USB2PHY1PLLPTUNE[3:0]	sysconf_usb2phy1_pllptune[3:0]	in	USB2PHY1 PLL proportional path adjustment
USB2PHY1PLLITUNE[1:0]	sysconf_usb2phy1_pllitune[1:0]	in	USB2PHY1 PLL integral path adjustment
USB2PHY1TXPREEMPPULSETUNE	sysconf_usb2phy1_txpreemppulsetune	in	USB2PHY1 HS transmitter pre-emphasis duration control
USB2PHY1TXPREEMPAMPTUNE[1:0]	sysconf_usb2phy1_txpreempamptune[1:0]	in	USB2PHY1 HS transmitter pre-emphasis current control
USBH_VBUSEN_POLARITY	sysconf_usb2_vbusen_polarity	in	Polarity of USBH_HS_VBUSEN alternate function
USBH_OVRCUR_POLARITY	sysconf_usb2_ovrcur_polarity	in	Polarity of USBH_HS_OVRCUR alternate function

73.3 USB2 OTG high-speed port

73.3.1 USB2 OTG port main features

- Can be configured statically as a host or a device (dual role)
- Dynamic switching between the 2 roles is not supported: host negotiation protocol (HNP) and session request protocol (SRP) are not supported.
- Embedded PHY which supports high-speed (HS), full speed (FS) and low speed (LS) modes (note: LS mode is not supported when configured as a device).
- Supports LPM (link power management)
- DMA engine with local 8 KBytes buffer
- Supports battery charging v1.2, with the exception of the accessory charger adapter mode
- Supports suspend, sleep, resume, and remote wakeup operations
- Provides electrical parameter override bits for optimal interoperability
- No dedicated VBUS ball connected to the USB2 PHY
- No dedicated ID ball connected to the USB2 PHY
- The supported standards are:
  - Universal serial bus specification, revision 2.0, USB implementers forum, Inc., April 27, 2000
  - Engineering change notice: USB2.0 link power management addendum, July 16, 2007
  - UTMI+ specification, revision 1.0, Philips semiconductor, February 25, 2004
  - Battery charging specification, revision 1.2, December 7, 2010
  - AMBA specification (revision 2.0, May 13, 1999)

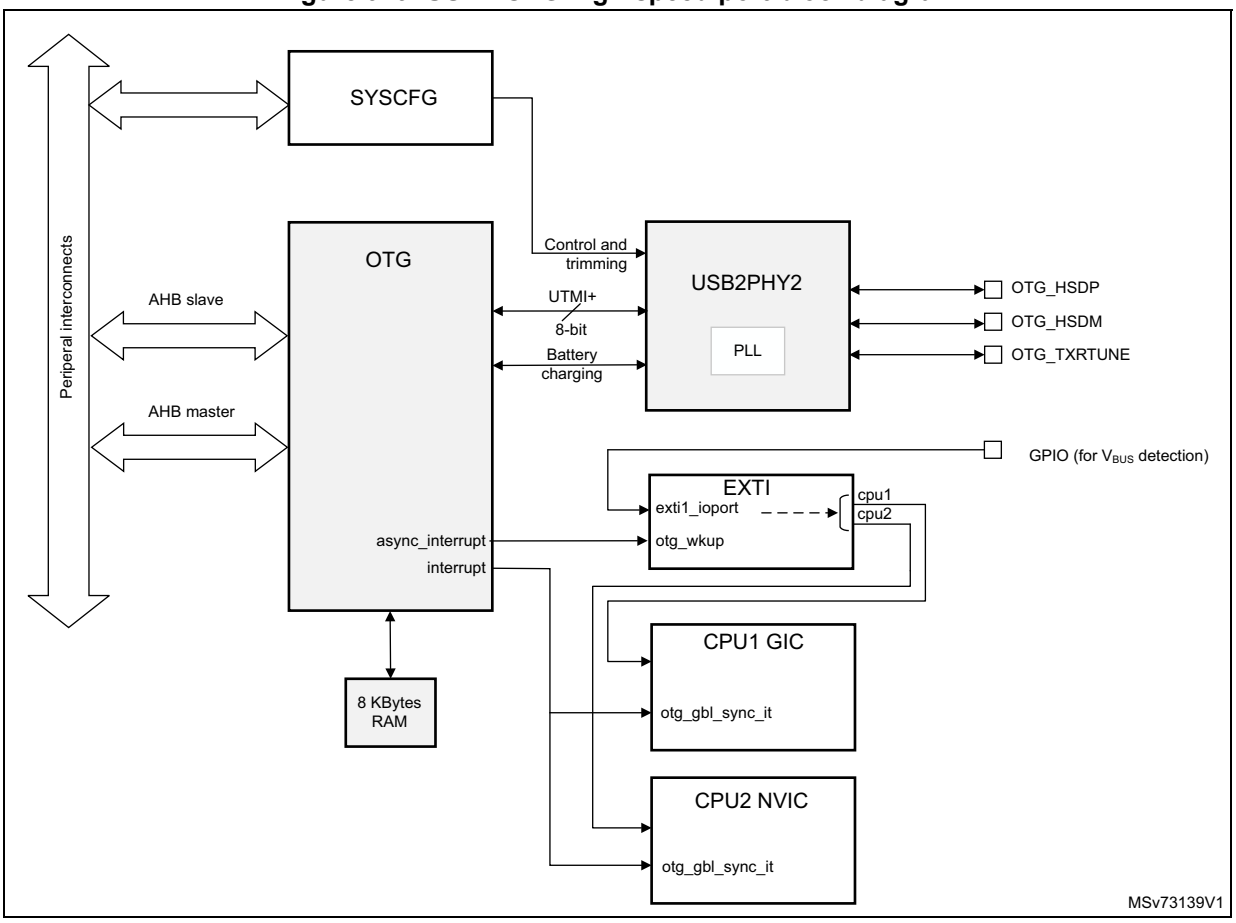
USB2 OTG high-speed port implementation

The OTG controller is based on Synopsys USB2 high-speed OTG designware core.

73.3.2 USB2 OTG high-speed port functional description

USB2 OTG high-speed port block diagram

Figure 919. USB2 OTG high-speed port block diagram



The AHB slave port of OTG is protected by a RISUP instance, and the traffic from the AHB master port is filtered by a RIMU instance. Both RISUP and RIMU instances are instantiated at device top level.

USB2 OTG high-speed port pins and internal signals

Table 773. USB2 OTG high-speed port pins

Signal name	Signal type	Description
OTG_HSDP	3.3 V analog	USB D+ signal
OTG_HSDM	3.3 V analog	USB D- signal
OTG_TXRTUNE	1.8 V analog	Transmitter resistor tune pin This analog signal connects to the ground through an external 200 Ω ±1% resistor that adjusts the high-speed source impedance of USB2PHY2.

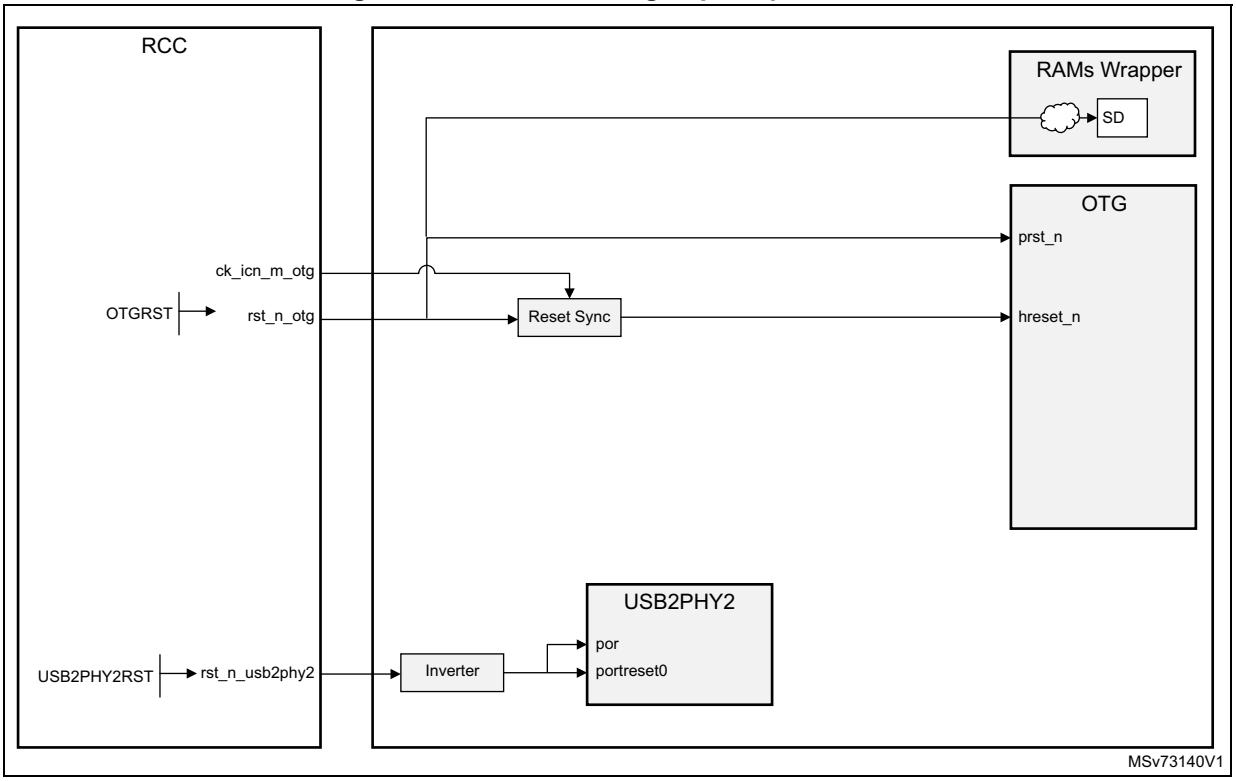
There are no dedicated alternate functions to enable an external V_{BUS} source or detect an overcurrent condition (relevant only when OTG is configured as a host). GPIOs must be used for this purpose.

Table 774. USB2 OTG high-speed port internal signals

Signal name	Signal type	Description
OTG AHB master	Digital	AHB master interface to the peripheral interconnects for memory access. AHBLite protocol, 32-bit data width.
OTG AHB slave	Digital	AHB slave interface for access to OTG registers. AHBLite protocol, 32-bit data width.

USB2 OTG high-speed port resets

Figure 920. USB2 OTG high-speed port resets



There are two resets for the USB2 OTG high-speed port:

- rst_n_otg which is controlled by the OTGRST bit in RCC_OTGCFGR register.
- rst_n_usb2phy2 which is controlled by the USB2PHY2RST bit in RCC_USB2PHY2CFGR register.

When configuring the USB2 OTG high-speed port, the reset sequence to be applied is:

1. rst_n_usb2phy2 and rst_n_otg are asserted
2. clocks coming from RCC and HSE are enabled
3. the frequency of USB2PHY2 reference clock is selected in SYSCFG_USB2PHY2CR register (it is assumed that the SYSCFG block has been taken out of reset previously)
4. USB2PHY2 trimming bits are programmed (if needed)
5. rst_n_usb2phy2 is released
6. wait for 50 µs (USB2PHY2 PLL lock time and start of generation of phyclock0)
7. rst_n_otg is released

USB2 OTG high-speed port clocking

Refer to *Figure 118: USBH and OTG ports clocking* in the RCC.

The OTG controller receives an AHB clock from the RCC and a processing clock from USB2PHY2. USB2PHY2 can receive its reference clock from either the RCC or directly from the HSE oscillator.

Table 775. OTG high-speed port clocks

IP	IP clock pin name	IP clock dir.	Sourced from	RCC clock line	Max Freq.	Description
OTG	hclk	in	RCC	ck_icn_m_otg	200 MHz	AHB clock
OTG	utmi_clk	in	USB2PHY2	-	60 MHz	Processing clock
USB2PHY2	CLKCORE	in	RCC	usb2phy2_ker_ck	24 MHz	Clock from RCC : one of the possible source of USB2PHY2 reference clock
USB2PHY2	CLKCORE	in	HSE	-	24 MHz	Clock from HSE oscillator : one of the possible source of USB2PHY2 reference clock
USB2PHY2	phyclock	out	-	-	60 MHz	UTMI 60 MHz clock

Details of the USB2 OTG high-speed port clocking

The USB2PHY2 receives its reference clock from the RCC or directly from the HSE oscillator. This is controlled by the bit USB2PHY2CKREFSEL in RCC_USB2PHY2CFGR register. If the USB2PHY2 reference clock is received from the RCC, it can be sourced from the HSE or re-generated by one of the PLLs and dividers located in the RCC.

The frequency of the USB2PHY2 reference clock can be selected among 19.2, 20 or 24 MHz (50 MHz is reserved for test purpose). This selection is controlled by the USB2PHY2SEL[2:0] bitfield in SYSCFG_USB2PHY2CR register. These bits must be set to the desired value before USB2PHY2 reset is released.

The 60 MHz processing clock provided by USB2PHY2 to OTG is obtained by division of the 480 MHz clock which is generated by a PLL inside the PHY (this is a fixed frequency PLL which cannot be programmed).

The USB2PHY2 PLL can be powered down during USB Suspend and Sleep modes. This is controlled by the USB2PHY2CMN bit in SYSCFG_USB2PHY2CR register.

The clocks provided by the RCC are controlled by enable bits in RCC_OTGCFGR and RCC_USB2PHY2CFGR registers.

USB2 OTG high-speed port memory requirements

Table 776. USB2 OTG high-speed port memory requirements

Port	Area required in the device memory map
OTG AHB slave	256 KBytes

Description of the USB2 OTG high-speed controller (OTG).

Refer to *Section 74: USB on-the-go high-speed (OTG)*.

Description of the USB2PHY2

Refer to *Section 23.2: SYSCFG registers*.

73.3.3 USB2 OTG high-speed port interrupts

The USB2 OTG high-speed port generates one interrupt and one wakeup event which can be used to take the system out of Stop mode.

Table 777. USB2 OTG high-speed port interrupt

USB OTG interrupt	IP pin name	CPU1 interrupt	CPU2 interrupt
OTG_GBL_SYNC_IT	OTG/interrupt	OTG_GBL_SYNC_IT	OTG_GBL_SYNC_IT

Table 778. USB2 OTG high-speed port wake-up event

USB OTG event	IP pin name	EXTI event
OTG_WKUP	OTG/async_interrupt	OTG_WKUP

In a 5 V device platform, a GPIO with an external resistor divider can be connected to VBUS and be used to detect its presence. This GPIO must be connected to EXTI and act as a wakeup event to take the system out of Stop Mode. Another option is to wakeup periodically the system from Stop mode and detect the presence of V_{BUS} with an ADC channel.

73.3.4 USB2PHY2 integration

In addition to the selection of the USB2PHY2 reference clock frequency, some SYSCFG register bits are used to control the trimming of the electrical parameters of the USB2PHY2.

The battery charging protocol relies on command and status bits implemented in the OTG_GCCFG register of the Synopsys USB2 high-speed OTG designware core.

Battery charging

The USB2PHY2 integrates the voltage and current sources, as well as the voltage comparators required to support battery charging v1.2. The relevant control bits are implemented in *OTG general core configuration register (OTG_GCCFG)*.

The control sequence to be applied depends on whether the USB OTG HS port is host or device.

Host case

If the application wants the host port to behave as a standard downstream port (SDP), the default values of H_CDPEN, H_CDPDETEN, H_VDMSRCEN bits in GGPIO register must be kept.

If the application wants the host port to behave as a charging downstream host (CDP), one of the following two control sequences must be implemented:

Control sequence 1:

- Set the H_CDPEN bit to 1
- Decode the disconnect information transmitted by the USB2PHY2 to the OTG controller (HOSTDISCONNECT0 output of USB2PHY2)
  - If disconnect = 1, set the H_VDMSRCEN bit to 1
  - Else set the H_VDMSRCEN bit to 0

Control sequence 2:

- Set the H_CDPEN bit to 1 and the H_CDPDETEN bit to 1
- Monitor CHGDET and FSVPLUS status bits:
  - If CHGDET = 1 and FSVPLUS = 0, set the H_VDMSRCEN bit to 1
  - Else, set the H_VDMSRCEN bit to 0

Device case

- Data contact detection
  - Set the PD_DCDET bit to 1. This enables a current source on OTG_HSDP and pulls OTG_HSDM low.
  - If FSVPLUS is at 0 for more than tDCD_DBNC (as defined in battery charging v1.2), data contact has happened. PD_DCDET must be reset.
- Primary detection
  - Set the PD_PDET bit to 1. This enables a voltage source on OTG_HSDP and a voltage comparator on OTG_HSDM.
  - If CHGDET = 0, the device is connected to a standard downstream port (SDP).
  - Else if CHGDET = 1, the device is connected to a charging downstream port (CDP) or a dedicated charging port (DCP).
  - Reset PD_PDET.
- Secondary detection
  - Set the PD_SDET bit to 1. This enables a voltage source on OTG_HSDM and a voltage comparator on OTG_HSDP.
  - If CHGDET = 0, the device is connected to a charging downstream port (CDP).
  - Else if CHGDET = 1, the device is connected to a dedicated charging port (DCP).
  - Reset PD_SDET.

Trimming of USB2PHY2 electrical parameters

Some parameters inside USB2PHY2 can be adjusted by control bits in SYSCFG_USB2PHY2TRIM1CR and SYSCFG_USB2PHY2TRIM2CR registers. These adjustments must be programmed while the USB2PHY2 is under reset.

Adjustments apply to: the internal PLL, the disconnect threshold, the squelch threshold, the data detection threshold, the transmitter high-speed crossover point, the FS/LS source impedance, the HS DC voltage level, the HS transmitter rise/fall times, the driver source impedance, the HS transmitter pre-emphasis current, the HS transmitter pre-emphasis duration.

The default values in SYSCFG_USB2PHY2TRIM1CR and SYSCFG_USB2PHY2TRIM2CR registers correspond to the recommended values.

ID management

There is no dedicated ID ball. For platforms with a micro-AB connector, a GPIO can be used to capture the ID information and depending on this information the software configures the controller in either host or device mode.

V_{BUS} detection

There is no dedicated VBUS ball. The V_{BUS} level comparators inside USB2PHY2 can therefore not be used for V_{BUS} detection. USB2PHY2 is configured in V_{BUS} external comparator mode by forcing VBUSVLDEXTSEL0 input of the PHY to 1. In V_{BUS} external comparator mode, V_{BUS} presence must be inferred from an external voltage comparator or by using a GPIO. This GPIO must be connected to the device event controller to be able to wake-up the system from Stop mode upon V_{BUS} detection.

It is also feasible to wake-up the system from Standby if V_{BUS} detection is managed externally and is connected to a wakeup pin of the device.

V_{BUS} enabling and over-current detection

There is no dedicated alternate function to enable an external V_{BUS} source in a SRC or DRP platform. Any GPIO configured as an output can be used to perform this function.

Also, there is no dedicated alternate function to capture an overcurrent information provided by an external V_{BUS} source. Any GPIO configured as an input can be used to perform this function.

73.3.5 SYSCFG registers

- *SYSCFG USB2PHY2 control register (SYSCFG_USB2PHY2CR)*
- *SYSCFG USB2PHY2 trimming 1 control register (SYSCFG_USB2PHY2TRIM1CR)*
- *SYSCFG USB2PHY2 trimming 2 control register (SYSCFG_USB2PHY2TRIM2CR)*

73.3.6 Connections to SYSCFG block

Table 779. Connections to SYSCFG block

SYSCFG bitfield	HSIF cluster pin name	HSIF dir.	Description
USB2PHY2SEL[2:0]	sysconf_usb2phy2_fsel[2:0]	in	Selection of USB2PHY2 reference clock frequency
USB2PHY2CMN	sysconf_usb2phy2_cmn	in	Control of the power down of USB2PHY2 analog blocks during Suspend and Sleep
AUTORSMENB2	sysconf_usb2phy2_autorsmemb_i	in	Enable of USB2PHY2 auto resume mode
RETENABLEN2	sysconf_usb2phy2_retenablen_i	in	Enable of USB2PHY2 retention mode
USB2PHY2TXRESTUNE[1:0]	sysconf_usb2phy2_txrestune[1:0]	in	USB2PHY2 source impedance adjustment
USB2PHY2TXRISETUNE[1:0]	sysconf_usb2phy2_txrisetune[1:0]	in	USB2PHY2 HS transmitter rise/fall time adjustment
USB2PHY2TXVREFTUNE[3:0]	sysconf_usb2phy2_txvreftune[3:0]	in	USB2PHY2 HS DC voltage level adjustment
USB2PHY2TXFSLSTUNE[3:0]	sysconf_usb2phy2_txfslstune[3:0]	in	USB2PHY2 FS/LS source impedance adjustment
USB2PHY2TXHSXVTUNE[1:0]	sysconf_usb2phy2_txhsxvtune [1:0]	in	USB2PHY2 transmitter HS crossover adjustment
USB2PHY2OTGTUNE[2:0]	sysconf_usb2phy2_otgtune[2:0]	in	USB2PHY2 V_{BUS} valid threshold adjustment
USB2PHY2VDATREFTUNE[1:0]	sysconf_usb2phy2_vdatreftune [1:0]	in	USB2PHY2 data detect voltage adjustment
USB2PHY2SQRXTUNE[2:0]	sysconf_usb2phy2_sqrxtune[2:0]	in	USB2PHY2 squelch threshold adjustment
USB2PHY2COMPDISTUNE[2:0]	sysconf_usb2phy2_compdistune [2:0]	in	USB2PHY2 disconnect threshold adjustment
USB2PHY2PLLPTUNE[3:0]	sysconf_usb2phy2_pllptune[3:0]	in	USB2PHY2 PLL proportional path adjustment
USB2PHY2PLLITUNE[1:0]	sysconf_usb2phy2_pllitune[1:0]	in	USB2PHY2 PLL integral path adjustment
USB2PHY2TXPREEMPPULSE TUNE	sysconf_usb2phy2_txpreemp pulsetune	in	USB2PHY2 HS transmitter pre-emphasis duration control
USB2PHY2TXPREEMPAMP TUNE[1:0]	sysconf_usb2phy2_txpreempamp tune[1:0]	in	USB2PHY2 HS transmitter pre-emphasis current control

73.4 Debug signals

Signals of the USB host high-speed host and the USB OTG high-speed ports are multiplexed and routed to the device hardware debug host (HDP). The SELUSBDBG bit of the SYSCFG peripheral controls the multiplexer, see *SYSCFG HDP USB debug control register (SYSCFG_HDPUSBDBGCR)*.

These debug signals are tapped on the UTMI+ interface between the controllers (USBH, OTG) and the PHYs (USB2PHY1, USB2PHY2). The list of debug signals is described in the table below.

Figure 921. Multiplexing of debug signals in the USB Sub-system

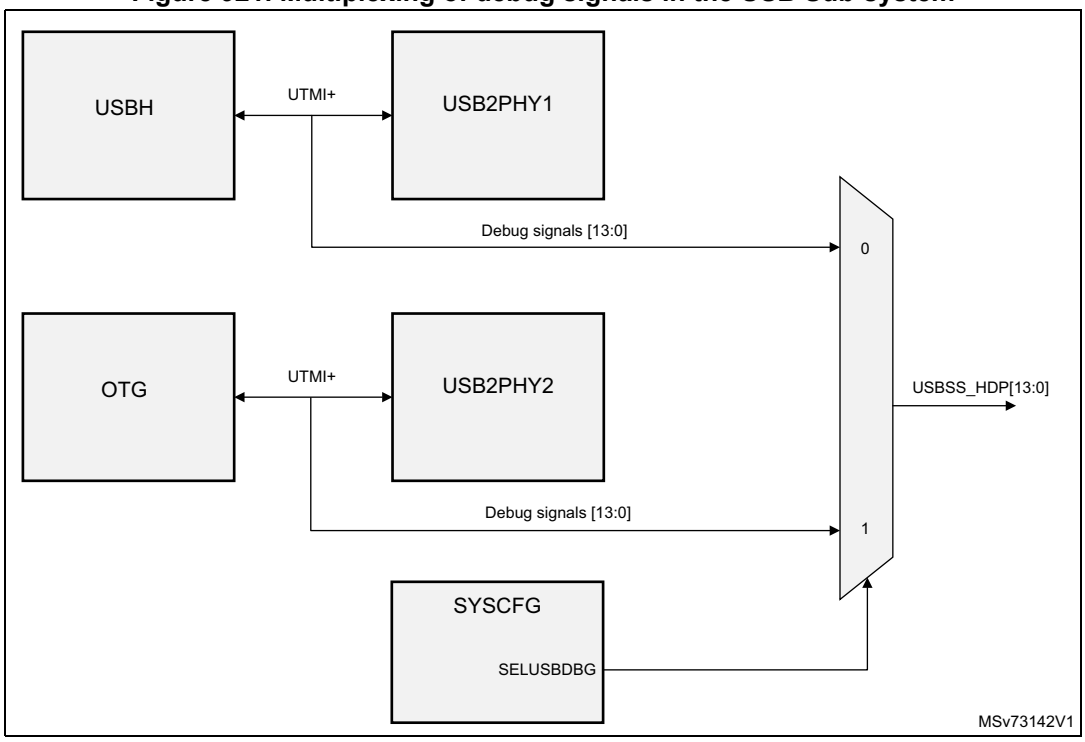


Table 780. USB debug signals

USBSS_HDP	USB host port	USB OTG port
[0]	suspendm0	suspendm0
[1]	sleepm0	sleepm0
[2]	linestate0[0]	linestate0[0]
[3]	linestate0[1]	linestate0[1]
[4]	hostdisconnect0	hostdisconnect0
[5]	0	iddig0
[6]	opmode0[0]	opmode0[0]
[7]	opmode0[1]	opmode0[1]
[8]	termsel0	termsel0

Table 780. USB debug signals (continued)

[9]	xcvrsel0[0]	xcvrsel0[0]
[10]	xcvrsel0[1]	xcvrsel0[1]
[11]	rxactive0	rxactive0
[12]	rxvalid0	rxvalid0
[13]	txvalid0	txvalid0

74 USB on-the-go high-speed (OTG)

74.1 OTG introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
HS	High-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG supports the speeds defined in the *Table 781: OTG speeds supported* below. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 781. OTG speeds supported

-	HS (480 Mbit/s)	FS (12 Mbit/s)	LS (1.5 Mbit/s)
Host mode	X	X	X
Device mode	X	X	-

74.2 OTG main features

The main features can be divided into three categories: general, host-mode, and device-mode features.

74.2.1 General features

The OTG interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0.
- OTG supports the following PHY interfaces:
  - A UTMI interface for internal HS PHY
- It includes support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications.
- It is software-configurable to operate as:
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG embeds an internal DMA with thresholding support and software selectable AHB burst type in DMA mode.
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 8 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets.
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.

74.2.2 Host-mode features

The OTG interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

74.2.3 Peripheral-mode features

The OTG interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

74.3 OTG implementation

Table 782. OTG implementation(1)

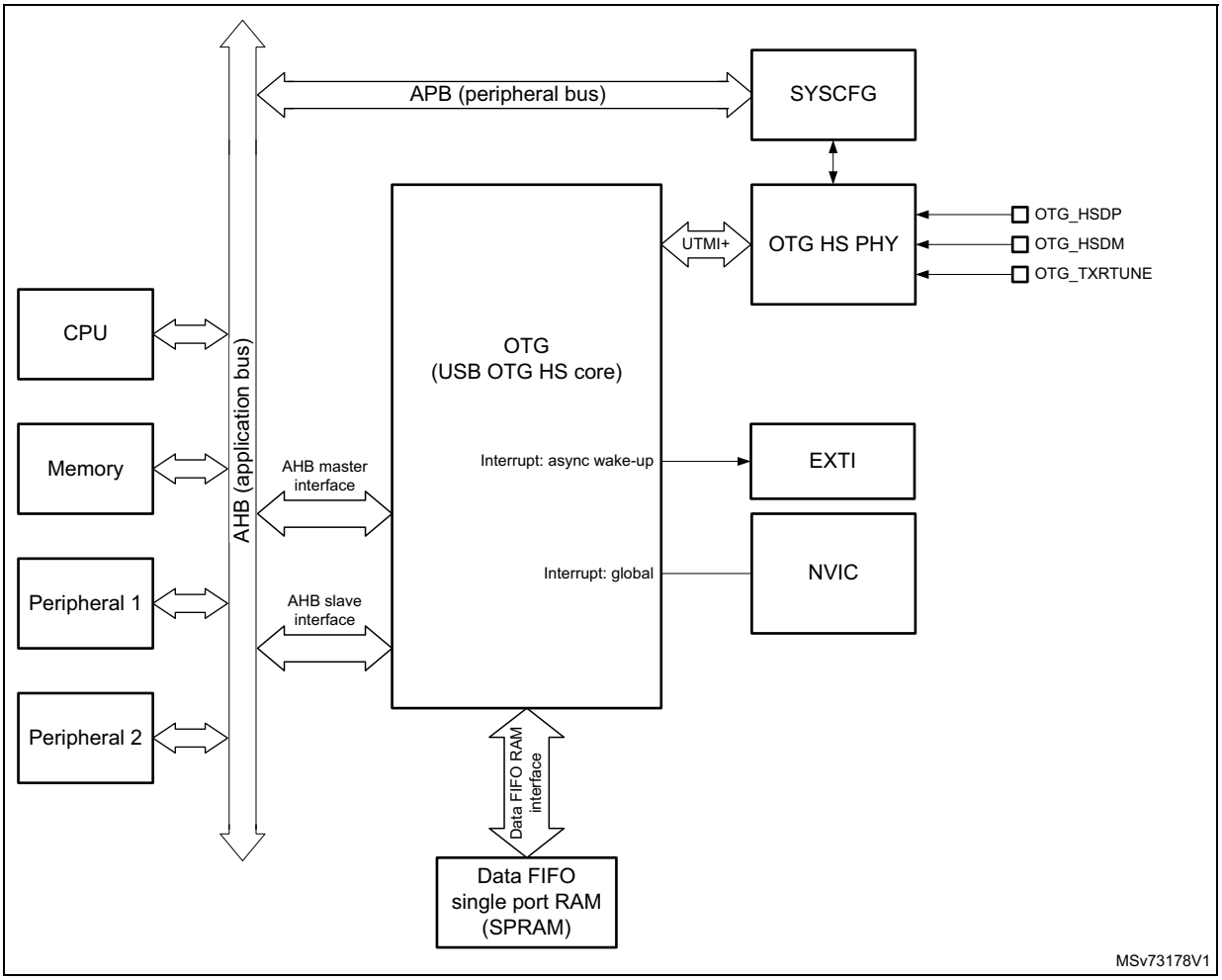
USB features	OTG
Device bidirectional endpoints (including EP0)	9
Host mode channels	16
Size of dedicated SRAM	8 Kbytes
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Battery charging detection (BCD) support	X
Integrated PHY	HS

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.

74.4 OTG functional description

74.4.1 OTG block diagram

Figure 922. OTG high-speed block diagram



74.4.2 OTG pin and internal signals

Table 783. OTG input/output pins

Signal name	Signal type	Description
OTG_HSDP	Digital input/output	USB OTG D+ line
OTG_HSDM	Digital input/output	USB OTG D- line
OTGx_TXRTUNE	Analog input	USB OTG Tx tune resistor

Table 784. OTG input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wake-up event output
usb_gbl_it	Digital output	USB OTG global interrupt

74.4.3 OTG core

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 74.12: OTG interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 8-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG peripheral, the AHB frequency must be higher than 30 MHz.

74.4.4 High-speed OTG PHY connected to OTG

Note: Refer to implementation table to determine if an HS PHY is embedded.

The USB OTG core includes an internal UTMI interface which is connected to the embedded HS PHY (see *Section 74.4.1: OTG block diagram*).

74.4.5 Battery charging detection

Support is available for the USB battery charging specification (v1.2).

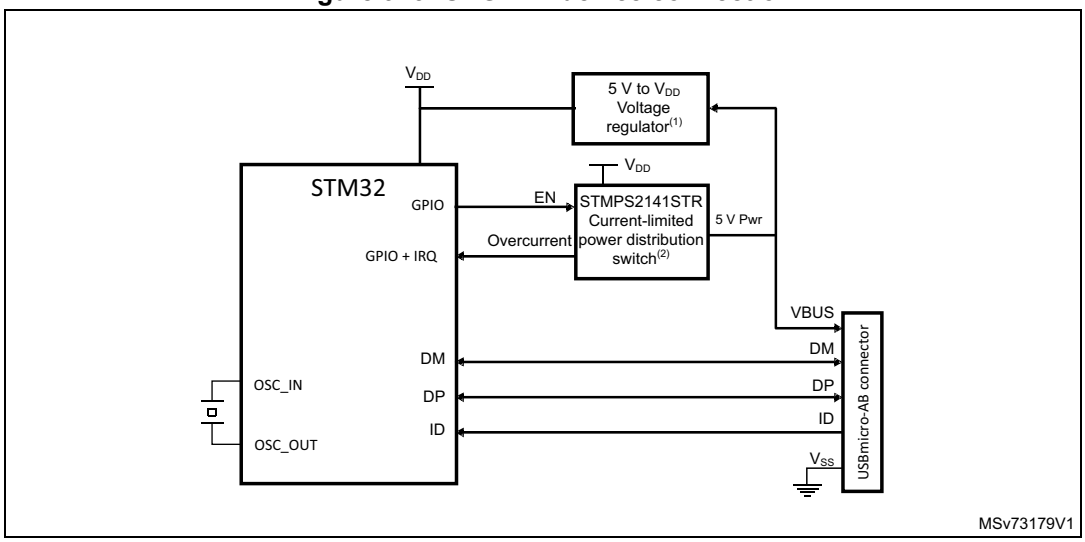
The OTG includes the necessary circuits to enable battery charging detection (portable device).

There is also a possibility to behave as a charging data port (the port advertises 1.5 A available on it's VBUS pin).

Various fields control this functions and are implemented in register OTG_GCCFG.

74.5 OTG dual role device (DRD)

Figure 923. OTG A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

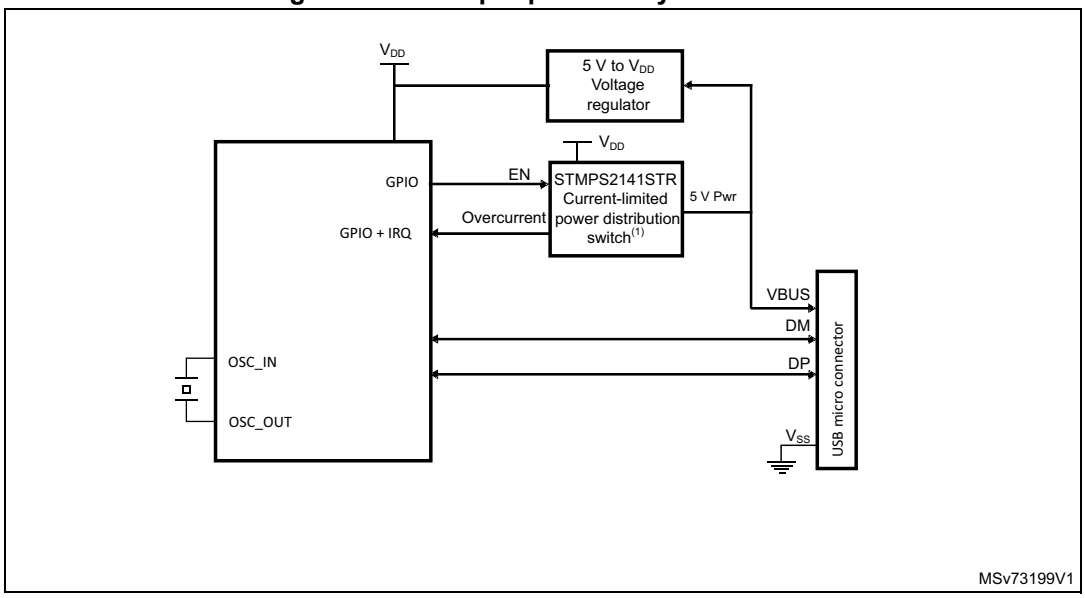
74.6 OTG as a USB peripheral

This section gives the functional description of the OTG in the USB peripheral mode. The OTG works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable.
- Peripheral only
  - The force device mode bit (FDMOD) in the *Section 74.14.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

Figure 924. OTG peripheral-only connection



1. Use a regulator to build a bus-powered device.

74.6.1 Peripheral states

Powered state

External circuitry should be used to detect presence of a voltage on the VBUS connector pin. Refer to AN4879 "Introduction to USB hardware and PCB guidelines using STM32 MCUs" for recommended components. The B-peripheral V_{BUS} detection should be overridden with OTG_GOTGCTL.BVALOEN = 1. The result of this detection should be maintained by writing the current value to OTG_GOTGCTL.BVALOVAL.

In the powered state, the OTG expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wake-up signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

74.6.2 Peripheral endpoints

The OTG core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Dedicated control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- eight IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has dedicated control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).
- eight OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a dedicated control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - endpoint enable/disable
  - endpoint activate in current configuration
  - program USB transfer type (isochronous, bulk, interrupt)
  - program supported packet size
  - program Tx FIFO number associated with the IN endpoint
  - program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - optionally program the STALL bit to always stall host tokens to that endpoint
  - optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG core updates them with the current transfer status.

The following transfer parameters can be programmed:

- transfer size in bytes
- number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- setup stage has been done (control-out only)
- associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- babble error condition has been detected
- endpoint disable by application is effective
- endpoint NAK by application is effective (isochronous-in only)
- more than 3 back-to-back setup packets were received (control-out only)
- timeout condition detected (control-in only)
- isochronous out packet has been dropped, without generating an interrupt

74.7 OTG as a USB host

This section gives the functional description of the OTG in the USB host mode. The OTG works as a USB host in the following circumstances:

Automatic host mode direct from ID pin:

The ID pin is not always available, refer to product datasheet for available pins.

- Use cases:
  - Micro connector: A-Device with a Micro-A plug inserted, and connector ID pin connected to the STM32 ID pin which is then detected in a low state.
- Note that the integrated pull-down resistors are automatically set on the DP/DM lines.

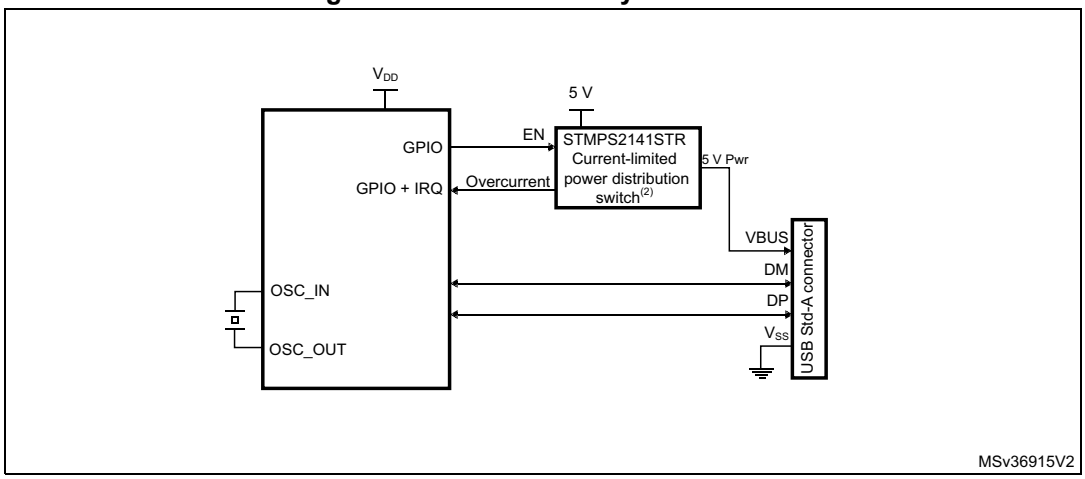
Manual forcing of host mode when not possible via ID pin:

The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG core to work as a USB host-only when required.

- Use cases:
  - Standard-A connector which always implies Host mode.
  - Micro connector, so ID present on connector, but ID pin not available in pinout, at A-plug insertion (detected by another means, for example: GPIO).
  - A Type-C connector, when host functionality is determined at cable attachment or via power delivery messages.
- Note that the integrated pull-down resistors are set on the DP/DM lines by setting bit FORCEHOSTPD.

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

Figure 925. OTG host-only connection



1. V_{DD} range is between 2 V and 3.6 V.

74.7.1 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

USB peripherals or B-device are detected as soon as they are connected. The OTG core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device's initiative (remote wake-up). In this case the remote wake-up interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wake-up signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

74.7.2 Host channels

The OTG core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of dedicated control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

74.7.3 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the

requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

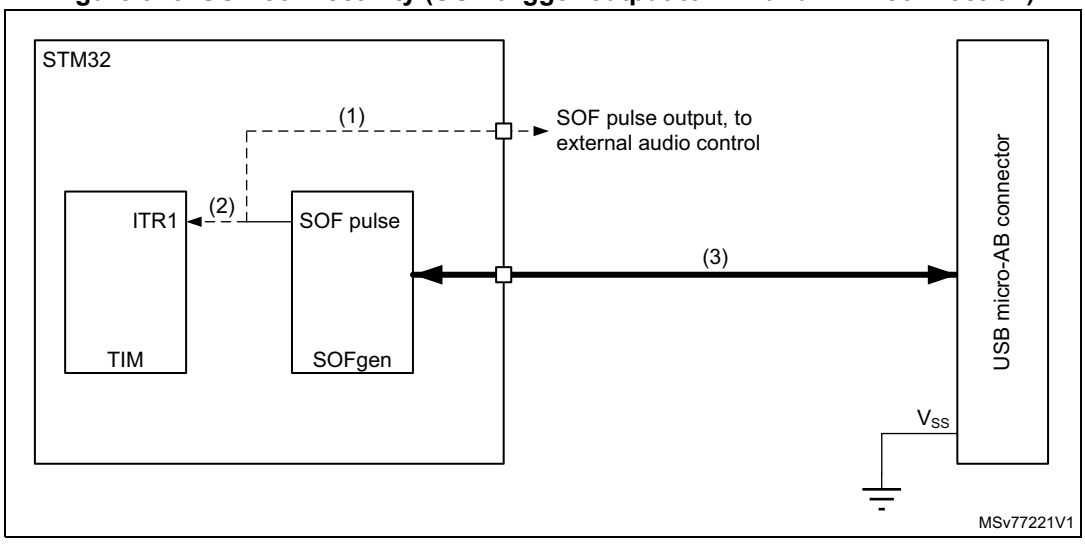
- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of eight entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of eight pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

74.8 OTG SOF trigger

Figure 926. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



1. SOF pulse output is not available on this product.
2. SOF timer connection is not available on this product.
3. See *Table 783: OTG input/output pins* for full pin list.

The OTG core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

74.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles.

74.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

74.9 OTG low-power modes

Table 785 below defines the STM32 low power modes and their compatibility with the OTG.

Table 785. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to *Section 19: Power control (PWR)* to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the transceiver is disabled, and only the part in charge of detecting the asynchronous resume or remote wake-up event is kept alive.

- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- USB system stop

  When the OTG is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG core automatically reactivates both system and USB clocks by asynchronous detection of remote wake-up (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG core.

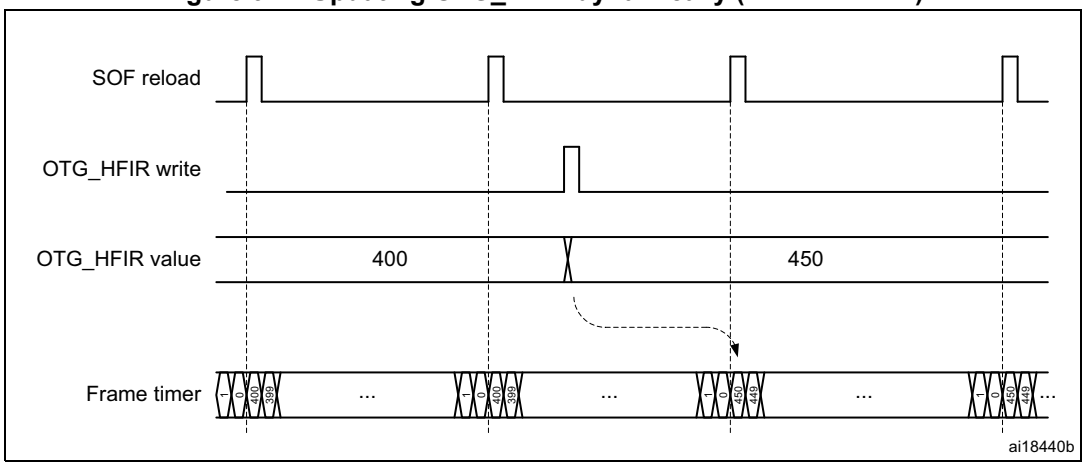
74.10 OTG Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in *Figure 927*.

For a dynamic update, it is required to set RLDCTRL=1.

Figure 927. Updating OTG_HFIR dynamically (RLDCTRL = 1)

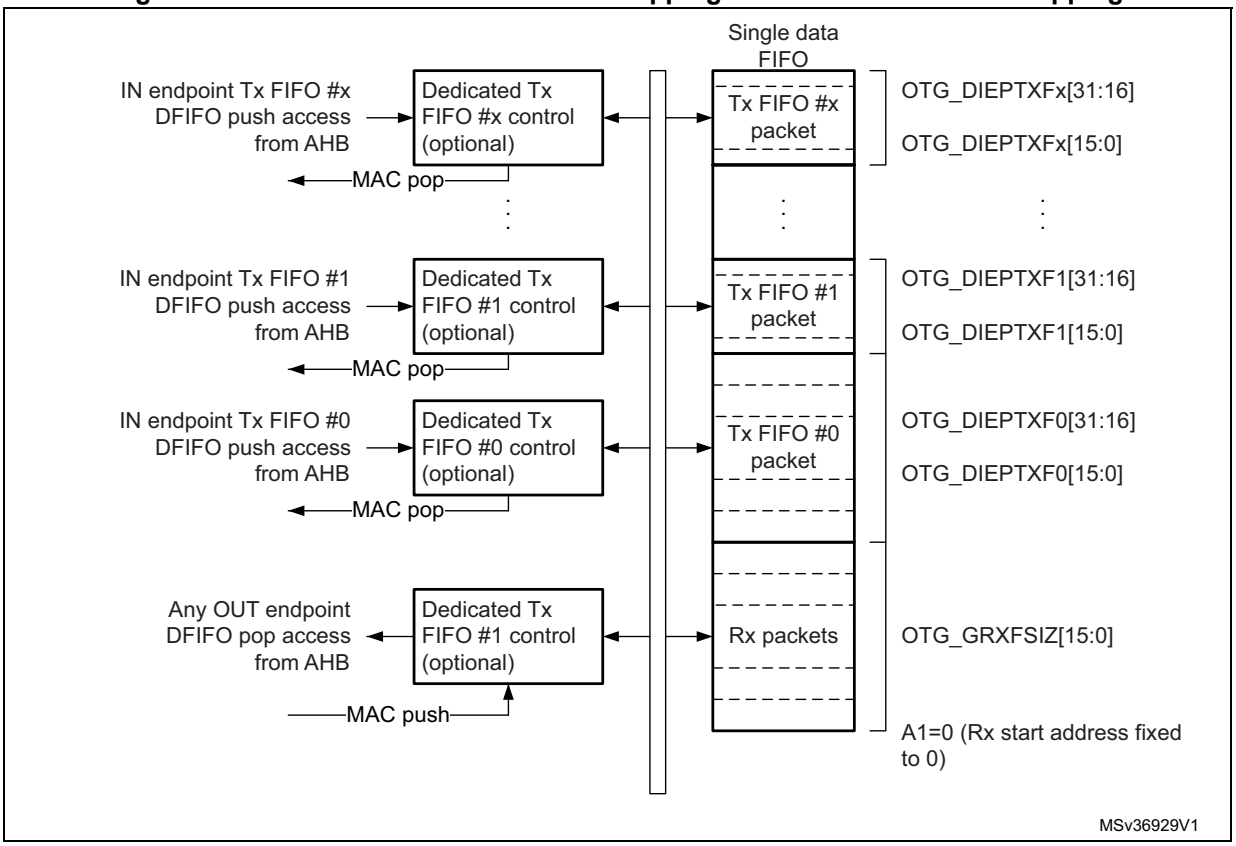


74.11 OTG data FIFOs

The USB system features 8 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

74.11.1 Peripheral FIFO architecture

Figure 928. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

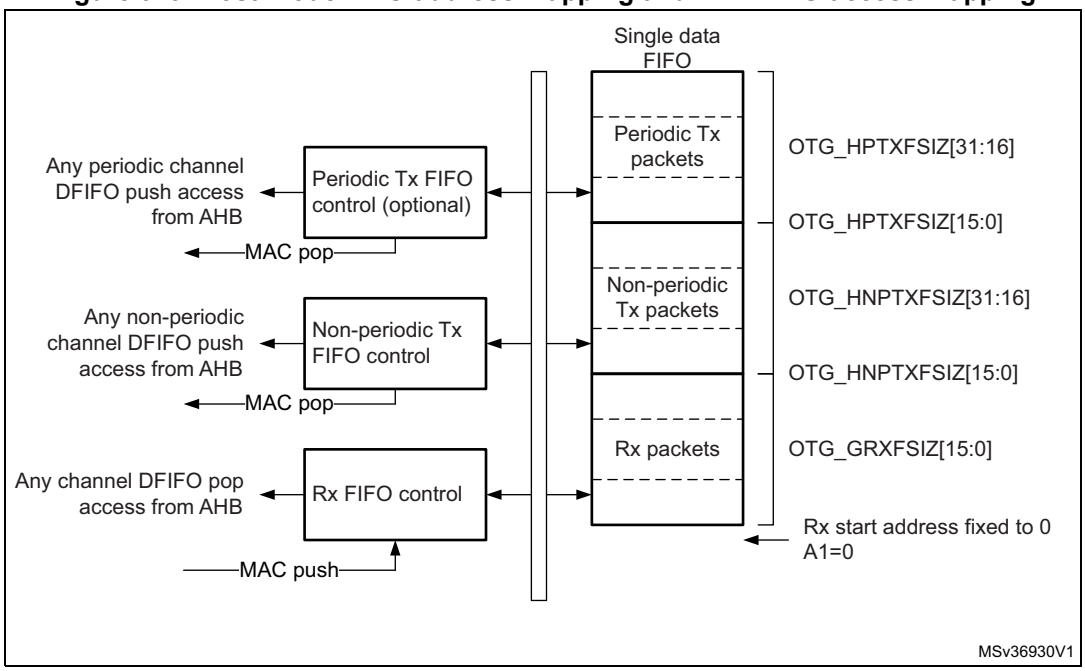
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

74.11.2 Host FIFO architecture

Figure 929. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

74.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application must allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

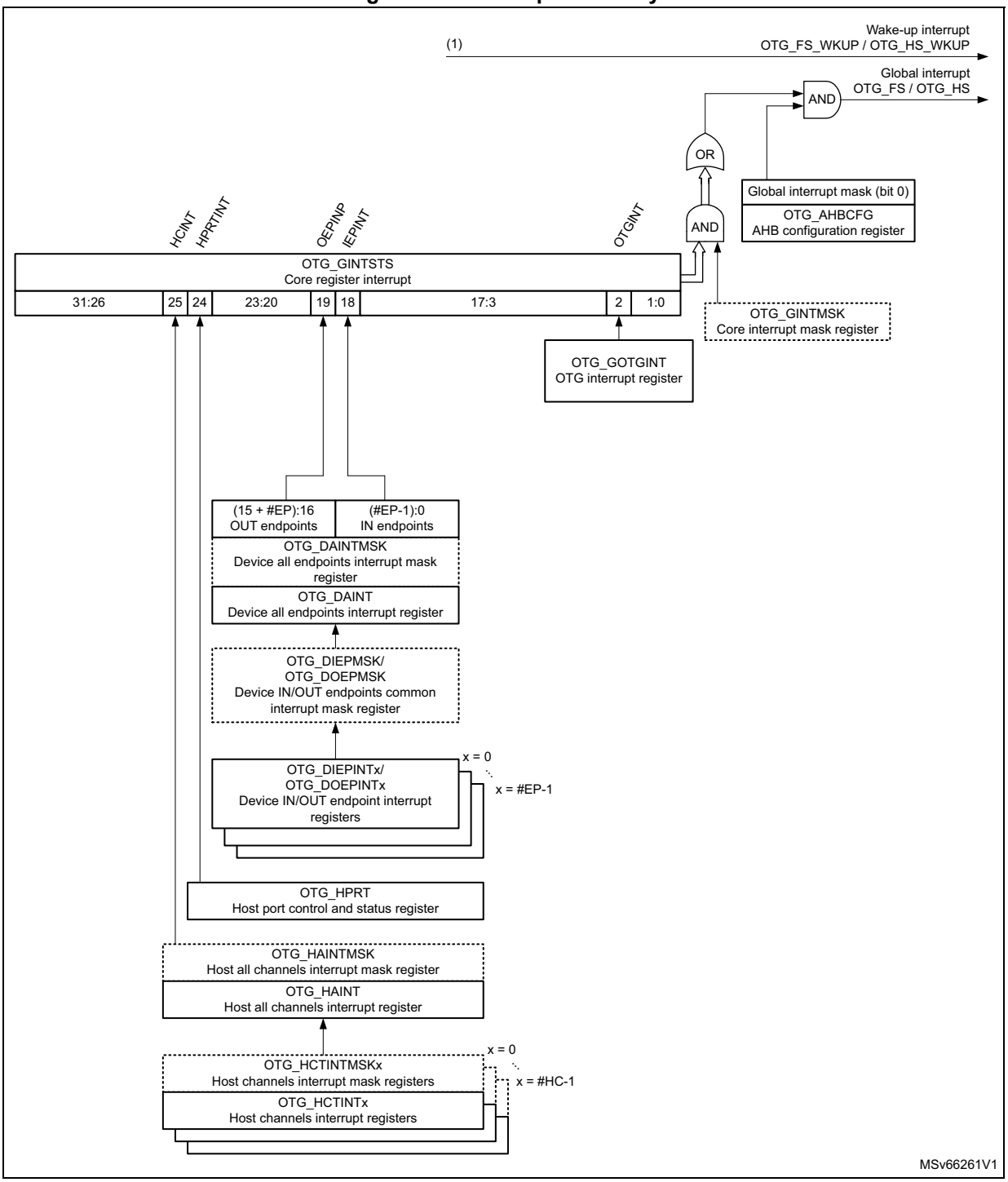
Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

74.12 OTG interrupts

When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 930 shows the interrupt hierarchy.

Figure 930. Interrupt hierarchy



1. OTG_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

74.13 OTG control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

74.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 786. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 74.14.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 74.14.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 74.14.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 74.14.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 74.14.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 74.14.6: OTG core interrupt register [alternate] (OTG_GINTSTS)* *Section 74.14.7: OTG core interrupt register [alternate] (OTG_GINTSTS)*

Table 786. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GINTMSK	0x018	*Section 74.14.8: OTG interrupt mask register [alternate] (OTG_GINTMSK)* *Section 74.14.9: OTG interrupt mask register [alternate] (OTG_GINTMSK)*
OTG_GRXSTSR	0x01C	*Section 74.14.10: OTG receive status debug read register [alternate] (OTG_GRXSTSR)* *Section 74.14.11: OTG receive status debug read register [alternate] (OTG_GRXSTSR)*
OTG_GRXSTSP	0x020	*Section 74.14.12: OTG status read and pop registers (OTG_GRXSTSP)* *Section 74.14.13: OTG status read and pop registers [alternate] (OTG_GRXSTSP)*
OTG_GRXFSIZ	0x024	*Section 74.14.14: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 74.14.15: OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)* *Section 74.14.16: Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 74.14.17: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 74.14.18: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 74.14.19: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 74.14.20: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 74.14.21: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x120	*Section 74.14.22: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 787. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 74.14.24: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 74.14.25: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 74.14.26: OTG host frame number/frame time remaining register (OTG_HFNUM)*
OTG_HPTXSTS	0x410	*Section 74.14.27: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*

Table 787. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HAINT	0x414	*Section 74.14.28: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 74.14.29: OTG host all channels interrupt mask register (OTG_HAINTMSK)*
OTG_HPRT	0x440	*Section 74.14.30: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x6E0	*Section 74.14.31: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCSPLTx	0x504 0x524 .... 0x6E4	*Section 74.14.32: OTG host channel x split control register (OTG_HCSPLTx)*
OTG_HCINTx	0x508 0x528 .... 0x6E8	*Section 74.14.33: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x6EC	*Section 74.14.34: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x6F0	*Section 74.14.35: OTG host channel x transfer size register (OTG_HCTSIZx)*
OTG_HCDMAx	0x514 0x534 .... 0x6F4	*Section 74.14.36: OTG host channel x DMA address register(OTG_HCDMAx)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 788. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 74.14.38: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 74.14.39: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 74.14.40: OTG device status register (OTG_DSTS)*

Table 788. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DIEPMSK	0x810	*Section 74.14.41: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 74.14.42: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*
OTG_DAINT	0x818	*Section 74.14.43: OTG device all endpoints interrupt register (OTG_DAINT)*
OTG_DAINTMSK	0x81C	*Section 74.14.44: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DTHRCTL	0x830	*Section 74.14.45: OTG device threshold control register (OTG_DTHRCTL)*
OTG_DIEPEMPMSK	0x834	*Section 74.14.46: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DIEPCTLx	0x900 0x920 ... 0xA00	*Section 74.14.47: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)* *Section 74.14.48: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 ... 0x9E8	*Section 74.14.49: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 74.14.50: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DIEPDMAx	0x914 0x934 ... 0x9F4	*Section 74.14.51: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)*
OTG_DTXFSTSx	0x918 0x938 ..... 0x9F8	*Section 74.14.52: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*
OTG_DIEPTSIZx	0x930 0x950 ... 0x9F0	*Section 74.14.53: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 74.14.54: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0XB28 ... 0xC08	*Section 74.14.55: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*

Table 788. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DOEPTSIZ0	0xB10	*Section 74.14.56: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPDMAx	0xB14 0xB34 ... 0xC14	*Section 74.14.57: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xC00	*Section 74.14.58: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)* *Section 74.14.59: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 .. 0xBF0	*Section 74.14.60: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 789. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x(1)/Host OUT Channel x(1): DFIFO write access Device OUT endpoint x(1)/Host IN Channel x(1): DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 8 in device mode and 15 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 790. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 74.14.61: OTG power and clock gating control register (OTG_PCGCCTL)*

74.14 OTG registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

74.14.1 OTG control and status register (OTG_GOTGCTL)

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

Address offset: 0x000

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	Res.	Res.	Res.	Res.	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	Res.	Res.
			rw					rw	rw	rw	rw	rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD:** Current mode of operation

Indicates the current mode (host or device).

0: Device mode

1: Host mode

Bit 20 **OTGVER:** OTG version

Selects the OTG revision.

0:OTG Version 1.3. OTG1.3 is obsolete for new product development.

1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid

Indicates the device mode transceiver status.

0: B-session is not valid.

1: B-session is valid.

In OTG mode, the user can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid

Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid
Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time

Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)
Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status

Indicates the connector ID status on a connect event.
0: The OTG controller is in A-device mode
1: The OTG controller is in B-device mode
Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable

It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **BVALOVAL:** B-peripheral session valid override value

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
0: Bvalid value is '0' when BVALOEN = 1
1: Bvalid value is '1' when BVALOEN = 1
Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
0: B-peripheral session valid is '0'
1: Use bit BVALOVAL to indicate B-peripheral session valid
Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value

This bit is used to set override value for Avalid signal when AVALOEN bit is set.
0: Avalid value is '0' when AVALOEN = 1
1: Avalid value is '1' when AVALOEN = 1
Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable

This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
0: A-peripheral session valid is '0'
1: Use bit AVALOVAL to indicate A-peripheral session valid
Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value
This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.
0: vbusvalid value is '0' when VBVALOEN = 1
1: vbusvalid value is '1' when VBVALOEN = 1
Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable
This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.
0: VBUS valid is '0'
1: Use bit VBVALOVAL to indicate A-peripheral session valid
Note: Only accessible in host mode.

Bits 1:0 Reserved, must be kept at reset value.

74.14.2 OTG interrupt register (OTG_GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTO CHG	Res.	Res.
													rc_w1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
													rc_w1		

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ADTOCHG:** A-device timeout change
The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.
Note: Accessible in both device and host modes.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **SEDET:** Session end detected
The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when V_{BUS} < 0.8 V.
Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.

74.14.3 OTG AHB configuration register (OTG_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFE LVL	TXFE LVL	Res.	DMAE N	HBSTLEN[3:0]				GINT MSK
							rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

Condition: device mode

This bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

Condition: host mode

This bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.

Bit 5 **DMAEN:** DMA enabled

0: The core operates in slave mode

1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]:** Burst length/type

0000 Single: Bus transactions use single 32 bit accesses (not recommended)

0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the INCR AHB bus command)

0011 INCR4: Bus transactions target 4x 32 bit accesses

0101 INCR8: Bus transactions target 8x 32 bit accesses

0111 INCR16: Bus transactions based on 16x 32 bit accesses

Others: Reserved

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

74.14.4 OTG USB configuration register (OTG_GUSBCFG)

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Address offset: 0x00C

Reset value: 0x0000 1400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw							rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYL PC	Res.	TRDT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOCAL[2:0]		
rw		rw	rw	rw	rw								rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force device mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force host mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **TSDPS**: TermSel DLine pulsing selection

This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
0: Data line pulsing using utmi_txvalid (default)
1: Data line pulsing using utmi_termsel

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **PHYLPC:** PHY Low-power clock select

This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.
0: 480 MHz internal PLL clock
1: 48 MHz external clock
In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits allows to set the turnaround time in PHY clocks. They must be configured according to *Table 791: TRDT values*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 791. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
30	-	0x9

74.14.5 OTG reset register (OTG_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

Address offset: 0x010

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	DMAR EQ	CSRST DONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXF FLSH	RXF FLSH	Res.	FCRST	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs		rs	rs	rw

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bit 30 **DMAREQ:** DMA request signal enabled

This bit indicates that the DMA request is in progress. Used for debug.

Bit 29 **CSRSTDONE**: Core soft reset done

The core sets this bit when all the necessary logic is reset in the core. This bit is cleared by the application along with CSRST.

0: Core reset not done

1: Core reset done

Bits 28:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

Condition: host mode

00000: Non-periodic Tx FIFO flush

00001: Periodic Tx FIFO flush

10000: Flush all the transmit FIFOs

Condition: device mode

00000: Tx FIFO 0 flush

00001: Tx FIFO 1 flush

00010: Tx FIFO 2 flush

...

01111: Tx FIFO 15 flush

10000: Flush all the transmit FIFOs

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the (micro-)frame number counter inside the core. When the (micro-)frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

When application writes '1' to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.

Note: Only accessible in host mode.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Can be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

- GATEHCLK bit in OTG_PCGCCTL
- STPPCLK bit in OTG_PCGCCTL
- FSLSPCS bits in OTG_HCFG
- DSPD bit in OTG_DCFG
- SDIS bit in OTG_DCTL
- OTG_GCCFG register
- PHYLPC bit in OTG_GUSBCFG
- TSDPS bit in OTG_GUSBCFG

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. The application must clear this bit after checking CSRSTDONE (core soft reset done). The application must also check that AHBIDL = 1 (AHB master is IDLE) before starting any operation.

Typically, the software reset is used during software development and also when the PHY selection bits are dynamically changed in the USB configuration registers listed above. When the PHY is changed, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

74.14.6 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Host mode, see next section for Device mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	DATAF SUSP	IPXFR	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1	rc_w1	rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wake-up detected interrupt

Wake-up interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

74.14.7 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Device mode, see previous section for Host mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	DATAF SUSP	IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1	rc_w1	rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wake-up detected interrupt

Wake-up interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **INCOMPISOOUT:** Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

74.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Host mode, see next section for Device mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	Res.	Res.	IPXFRM	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	r			rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
										rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wake-up detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bits 20:6 Reserved, must be kept at reset value.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

74.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	Res.	CIDSCHGM	LPMINTM	Res.	Res.	Res.	RSTDETM	FSUSPM	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	Res.	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw		rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wake-up detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 22 **FSUSPM:** Data fetch suspended mask
0: Masked interrupt
1: Unmasked interrupt

Bit 21 **IISOOXFRM:** Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

74.14.10 OTG receive status debug read register [alternate] (OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

74.14.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

74.14.12 OTG status read and pop registers (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

74.14.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

74.14.14 OTG receive FIFO size register (OTG_GRXFSIZ)

The application can program the RAM size that must be allocated to the Rx FIFO.

Address offset: 0x024

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD[15:0]:** Rx FIFO depth
This value is in terms of 32-bit words.
Maximum value is 1024
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

74.14.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)

Valid for Host mode, see next section for Device mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

74.14.16 Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)

Valid for Device mode, see previous section for Host mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX0FD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TX0FD[15:0]:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA[15:0]:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

74.14.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

Address offset: 0x02C

Reset value: 0x0008 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
XXXX00X: IN/OUT token
XXXX01X: Zero-length transmit packet (device IN/host OUT)
XXXX11X: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

74.14.18 OTG general core configuration register (OTG_GCCFG)

This register is available in host and device modes.

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDPULLUPDIS	Res.	BCDEN	FORCEHOSTPD	Res.	VBVALOVAL	PD_SDET	Res.	PD_PDET	PD_DCDET	HVDMSRCEN	HCDPDETEN	HCDPEN
			rw		rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSVMINUS	FSVPLUS	CHGDET
													r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDPULLUPDIS:** ID pin pull-up disable
- 0: Enable ID pin pull-up
- 1: Disable ID pin pull-up

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BCDEN:** Force Battery charging (BC) mode

This bit is set by the software to force the Battery charging mode. When enabled, the USB PHY is fully controlled by Battery Charging controls and cannot be used for normal communication.
- 0: Normal USB mode
- 1: Battery Charging mode

Bit 25 **FORCEHOSTPD:** Force host mode pull-downs

If the ID pin functions are enabled, the host mode pull-downs on DP and DM activate automatically. However, whenever that is not the case, yet host mode is required, this bit must be used to force the pull-downs active.
- 0: Do not force host mode pull-downs
- 1: Force host mode pull-downs

Bit 24 Reserved, must be kept at reset value.

Bit 23 **VBVALOVAL:** Software override value of the VBUS B-session detection
- 0: B-session inactive
- 1: B-session active

Bit 22 **PD_SDET:** Battery charging specification – secondary detection enable
- 0: Secondary detection disabled
- 1: Secondary detection enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **PD_PDET:** Battery charging specification – primary detection enable
- 0: Primary detection disabled
- 1: Primary detection enabled

Bit 19 **PD_DCDET:** Battery charging specification – data contact detection enable
- 0: Data Contact Detection disabled
- 1: Data Contact Detection enabled

Bit 18 **HVDMSRCEN:** Battery charging specification – host CDP port voltage source enable on DM
0: DM voltage source disabled
1: DM Voltage source enabled

Bit 17 **HCDPDETEN:** Battery charging specification – host CDP port voltage detector enable on DP
0: DP voltage detection disabled
1: DP voltage detection enabled

Bit 16 **HCDPEN:** Battery charging specification – host CDP behavior enable
0: Disable CDP behavior
1: Enable CDP behavior

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FSVMINUS:** Battery charging specification – single-ended DM indicator
This bit gives the voltage level on DM (also result of the comparison with V_{LGC} threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 1 **FSVPLUS:** Battery charging specification – single-ended DP indicator
This bit gives the voltage level on DP (also result of the comparison with V_{LGC} threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 0 **CHGDET:** Battery charging specification – charger detection, result of the current mode (primary or secondary).
0: Low value on pin
1: High value on pin

74.14.19 OTG core ID register (OTG_CID)

This is a register containing the Product ID as reset value.

Address offset: 0x03C

Reset value: 0x0000 6100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field
Application-programmable ID field.

74.14.20 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL:** Enable best effort service latency

This bit enables the BESL feature as defined in the LPM errata:

0: The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007

1: The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007

Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit must be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status

Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]:** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).
This bit is reset when SLPSTS is 0.
1: The application or host can start resume from Sleep state
0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.
The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.
The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.
The core clears this bit after:
– The core detects a remote L1 wake-up signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/ remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).
0: Core not in L1
1: Core in L1

Bits 14:13 **LPMRSP[1:0]:** LPM response

Device mode:
The response of the core to LPM transaction received is reflected in these two bits.
Host mode:
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bits 11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

Host mode:

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (µs):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:

The value of remote wake up to be sent in the wIndex field of LPM transaction.

Device mode (read-only):

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

– No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
– Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
– No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

– The received bLinkState is not L1 (STALL response), or
– An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled
1: LPM capability is enabled

74.14.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0400 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

74.14.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 8)

Reset value: 0x0200 0400, 0x0200 0600, 0x0200 0800, 0x0200 0A00, 0x0200 0C00, 0x0200 0E00, 0x0200 1000, 0x0200 1200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

74.14.23 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

74.14.24 OTG host configuration register (OTG_HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

Condition: FS host mode
01: PHY clock is running at 48 MHz
Others: Reserved
Condition: LS host mode
00: Reserved
01: Select 48 MHz PHY clock frequency
10: Select 6 MHz PHY clock frequency
11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

74.14.25 OTG host frame interval register (OTG_HFIR)

This register stores the frame interval information for the current speed to which the OTG controller has enumerated.

Address offset: 0x404

Reset value: 0x0000 EA60

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR cannot be reloaded dynamically

1: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 125 µs × (FRIVL - 1) in high speed operation

– Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation

74.14.26 OTG host frame number/frame time remaining register (OTG_HFNUM)

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

Address offset: 0x408

Reset value: 0x0000 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTREM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

74.14.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

Address offset: 0x410

Reset value: 0x0008 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0XXXXXXX: send in even frame

1XXXXXXX: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

XXXXX00X: IN/OUT

XXXXX01X: Zero-length packet

XXXXX11X: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

0: Periodic transmit request queue is full

1: 1 location available

2: 2 locations available

n: n locations available (0 ≤ n ≤ 8)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0: Periodic Tx FIFO is full

1: 1 word available

2: 2 words available

n: n words available (where 0 ≤ n ≤ PTXFD)

Others: Reserved

74.14.28 OTG host all channels interrupt register (OTG_HAINT)

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 930*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

Address offset: 0x414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

74.14.29 OTG host all channels interrupt mask register (OTG_HAINTMSK)

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Address offset: 0x418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

74.14.30 OTG host port control and status register (OTG_HPRT)

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 930*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

Address offset: 0x440

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL[3]
													r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved
00: High speed

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition. Note that this bit does not directly activate the voltage on the connector VBUS.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_HSDP
Bit 11: Logic level of OTG_HSDM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wake-up signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wake-up sequence, as indicated by the port resume/remote wake-up detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wake-up sequence, as indicated by the port L1Resume/Remote L1wake-up detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

74.14.31 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.

0: OUT

1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

74.14.32 OTG host channel x split control register (OTG_HCSPLTx)

Address offset: 0x504 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLIT EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP LSPLT
rw															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XACTPOS[1:0]		HUBADDR[6:0]							PRTADDR[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPLITEN:** Split enable

The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT:** Do complete split

The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS[1:0]:** Transaction position

This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.

11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)

10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)

00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)

01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR[6:0]:** Hub address

This field holds the device address of the transaction translator hub.

Bits 6:0 **PRTADDR[6:0]:** Port address

This field is the port number of the recipient transaction translator.

74.14.33 OTG host channel x interrupt register (OTG_HCINTx)

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 930*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERR	FRM OR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHB ERR	CHH	XFRC
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR:** Data toggle error.

Bit 9 **FRMOR:** Frame overrun.

Bit 8 **BBERR:** Babble error.

Bit 7 **TXERR:** Transaction error.

Indicates one of the following errors occurred on the USB.

CRC check failure

Timeout

Bit stuff error

False EOP

Bit 6 **NYET:** Not yet ready response received interrupt.

Bit 5 **ACK:** ACK response received/transmitted interrupt.

Bit 4 **NAK:** NAK response received interrupt.

Bit 3 **STALL:** STALL response received interrupt.

Bit 2 **AHBERR:** AHB error

This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH:** Channel halted

This indicates that the transfer completed:

– because of any USB transaction error

– in response to disable request by the application

– normally

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

74.14.34 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

This register reflects the mask for each channel status described in the previous section.

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 6 **NYET:** response received interrupt mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error.
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

74.14.35 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO PNG	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping
This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.
Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]:** Data PID
The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.
00: DATA0
01: DATA2
10: DATA1
11: SETUP (control) / MDATA (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

74.14.36 OTG host channel x DMA address register(OTG_HCDMAx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA address

This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

74.14.37 Device-mode registers

These registers must be programmed every time the core changes to device mode

74.14.38 OTG device configuration register (OTG_DCFG)

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Address offset: 0x800

Reset value: 0x0220 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **PERSCHIVL[1:0]:** Periodic schedule interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame
01: 50% of (micro)frame
10: 75% of (micro)frame
11: Reserved

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.

1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

00: High speed

01: Full speed

10: Reserved

11: Reserved

74.14.39 OTG device control register (OTG_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.
													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL[2:0]			GONSTS	GINSTS	SDIS	RWUSIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wake-up from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control
000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status
0: A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1: No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status
0: A handshake is sent out based on the data availability in the transmit FIFO.
1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect
The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.
0: Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.
1: The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wake-up signaling
When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.
If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 792 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 792. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs
High speed	Not Idle or suspended (Performing transactions)	125 µs

74.14.40 OTG device status register (OTG_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

Address offset: 0x808

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSP STS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]:** Device line status

Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG controller has come up after speed detection through a chirp sequence.
00: High Speed
01: Full Speed
11: Reserved
Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines
– When the application writes to the remote wake-up signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

74.14.41 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x810

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw					rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

74.14.42 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x814

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	B2B STUPM	STS PHSR XM	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw				rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

74.14.43 OTG device all endpoints interrupt register (OTG_DAINT)

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

Address offset: 0x818

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

74.14.44 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

Address offset: 0x81C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

74.14.45 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN[8:0]									RXTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXTHRLEN[8:0]									ISOTHREN	NONISOTHREN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **ARPEN:** Arbiter parking enable
This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.

Bit 26 Reserved, must be kept at reset value.

Bits 25:17 **RXTHRLEN[8:0]:** Receive threshold length

This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable

When this bit is set, the core enables thresholding in the receive direction.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN[8:0]:** Transmit threshold length

This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable

When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable

When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

74.14.46 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

Address offset: 0x834

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits

These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

74.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID register fields to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

74.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

74.14.49 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 930*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTD RPSTS	Res.	Res.	TXFIF OUD RN	TXFE	IN EPNE	IN EPNM	ITTXFE	TOC	AHB ERR	EP DISD	XFRC
		rc_w1		rc_w1			rc_w1	r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFIFOUDRN:** Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

74.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

Address offset: 0x910

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		Res.	Res.	Res.
											rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

74.14.51 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

74.14.52 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available

Indicates the amount of free space available in the endpoint Tx FIFO.

Values are in terms of 32-bit words:

0x0: Endpoint Tx FIFO is full

0x1: 1 word available

0x2: 2 words available

0xn: n words available

Others: Reserved

74.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

74.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–8.

Address offset: 0xB00

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.
00: 64 bytes
01: 32 bytes
10: 16 bytes
11: 8 bytes

74.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 930*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPK TRX	NYET	NAK	BERR	Res.	Res.	Res.	OUT PKT ERR	Res.	B2B STUP	STSPH SRX	OTEP DIS	STUP	AHB ERR	EP DISD	XFRC
rc_w1	rc_w1	rc_w1	rc_w1				rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STPKTRX:** Setup packet received

Applicable for control OUT endpoints in only in the Buffer DMA Mode. Set by the OTG, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG can receive any number of back to back setup packets and one buffer for every setup packet is used.

Bit 14 **NYET:** NYET interrupt

This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt
The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERR**: OUT packet error
This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received
Applies to control OUT endpoint only.
This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write
This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled
Applies only to control OUT endpoints.
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done
Applies to control OUT endpoint only.Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 **AHBERR**: AHB error
This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt
This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC:** Transfer completed interrupt
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

74.14.56 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

Address offset: 0xB10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

74.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

74.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID
Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID register fields to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

74.14.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

74.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID[1:0]:**

Condition: isochronous OUT endpoints
Received data PID
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA
Condition: control OUT endpoints
STUPCNT[1:0]: SETUP packet count
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

74.14.61 OTG power and clock gating control register (OTG_PCGCCTL)

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wake-up logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

74.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1)

This register is available in host and device modes.

Address offset: 0xE04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAM GATE EN	CNTGATECLK [1:0]		GATE EN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RAMGATEEN:** Enable RAM clock gating

Enable gating of the FIFO RAM.

Bits 2:1 **CNTGATECLK[1:0]:** Counter for clock gating

Indicates to the controller how many PHY Clock cycles and AHB Clock cycles of 'IDLE' (no activity) the controller waits for before Gating the respective PHY and AHB clocks internal to the controller.

00: 64 clocks

01: 128 clocks

10: Reserved

11: Reserved

Bit 0 **GATEEN:** Enable active clock gating

The application programs GATEEN to enable Active Clock Gating feature for the PHY and AHB clocks.

74.14.63 OTG register map

The table below gives the USB OTG register map and reset values.

Table 793. OTG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_ GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	Res.	Res.	Res.	Res.	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	Res.	Res.
	Reset value											0	0	0	0	0	1				0					0	0	0	0	0	0		
0x004	OTG_ GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTOCHG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value														0								-								0		
0x008	OTG_ GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	DMAEN	HBSTLEN				GINTMSK
	Reset value																								0	0		0	0	0	0	0	0
0x00C	OTG_ GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.	PHYLPC	Res.	TRDT				Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOCAL		
	Reset value		0	0							0							0		0	1	0	1								0	0	0
0x010	OTG_GRSTCTL	AHBIDL	DMAREQ	CSRSTDONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res.	Res	Res.	Res	Res	Res	TXFNUM					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
	Reset value	1	0	0																			0	0	0	0	0	0	0		0	0	0
0x014	OTG_GINTSTS (Device mode)	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	DATAFSUSP	IPXFR	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
	OTG_GINTSTS (Host mode)	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	DATAFSUSP	INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	OTG_GINTMSK (Device mode)	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	
	OTG_GINTMSK (Host mode)	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	OTG_GRXSTSR (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_GRXSTSR (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_GRXSTSP (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_ GRXFSIZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_ HNPTXFSIZ (Device mode)	NPTXFD																NPTXFSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	OTG_ DIEPTXF0 (Host mode)	TX0FD																TX0FSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_ HNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	OTG_GCCFG	Res.	Res.	Res.	IDPULLUPDIS	Res.	FORCEHOSTPD	Res.	FORCEHOSTPD	PD_SDET	Res.	PD_PDET	PD_DCDET	HVDMSRCEN	HCDPDETEN	HCDPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSVMINUS	FSVPLUS	CHGDET
	Reset value				0		0	0		0	0		0	0	0	0	0														X	X	X

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C	**OTG_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
0x054	OTG_GLPMCFG	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS				L1SSEN	REMWAKE	BESL				LPMACK	LPMEN
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	OTG_HPTXFSIZ	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x104	OTG_ DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x120	OTG_DIEPTXF7	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_HCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLS PCS	
	Reset value																														0	0	0
0x404	OTG_HFIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTRL	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	OTG_HAINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_ HAINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_HPRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSP D	PTCTL					PPWR	PLSTS		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x500	OTG_ HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x504	OTG_ HCSPLT0	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_ HCINTMSK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	OTG_HCTSIZ0	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HCDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.Iterating preceding block of registers starting at offset 0x500..																															
0x6E0	OTG_ HCCHAR15	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E4	OTG_ HCSPLT15	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E8	OTG_HCINT15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x6EC	OTG_ HCINTMSK15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6F0	OTG_ HCTSIZ15	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F4	OTG_HCDMA15	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_DCFG	Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value							0	0									0	0		0	0	0	0	0	0	0	0		0	0	0
0x804	OTG_DCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0							0	0	0	0	0	0	0	0	0	1	0
0x808	OTG_DSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS		FNSOF														Res.	Res.	Res.	Res.	EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	OTG_DIEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.		TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0					0		0	0	0	0	0	0	0
0x814	OTG_DOEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	B2BSTUPM	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0				0		0		0	0	0	0	0
0x818	OTG_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_ DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x830	OTG_ DTHRCTL	Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN							RXTHREN	Res.	Res.	Res.	Res.	TXTHRLEN												ISOTHREN	NONISOTHREN
	Reset value					0									0							0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_DIE PEMPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x900	OTG_ DIEPCTL0 (INT/BULK)	EPENA	EPDIS	SD1PID	SD1P0D	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL0 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_ DIEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x910	OTG_ DIEPTSIZ0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x914	OTG_ DIEPDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	OTG_ DTXFSTS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0x900.																															
0x9E0	OTG_ DIEPCTL7 (INT/BULK)	EPENA	EPDIS	SD1PID	SD1P0D	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL7 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9E8	OTG_ DIEPINT7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x9F0	OTG_ DIEPTSIZ7	Res.	MCNT		PKTCNT									XFRSIZ																			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9F8	OTG_ DTXFSTS7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB00	OTG_ DOEPCTL0	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	OTG_ DOEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB10	OTG_ DOEPTSIZ0	Res.	STUP CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB14	OTG_ DOEPDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB20	OTG_ DOEPCTL1 (INT/BULK)	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL1 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	OTG_ DOEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB30	OTG_ DOEPTSIZ1	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	OTG_ DOEPDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0xB20.																															

Table 793. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC00	OTG_ DOEPCTL8 (INT/BULK)	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL8 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xC08	OTG_ DOEPINT8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0		-		0		0	0	0	0	0	0	0
0xC10	OTG_ DOEPTSIZ8	Res.	RXDPID/ STUPCNT		PKTCNT									XFRSIZ																			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC14	OTG_ DOEPDMA8	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_ PCGCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0
0xE04	OTG_ PCGCCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMGATEEN	CNTGATECLK		GATEEN
	Reset value																													0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

74.15 OTG programming model

74.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core's configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   – Global interrupt mask bit GINTMSK = 1
   – Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   – Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   – OTG timeout calibration field
   – USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:
   OTG interrupt mask
   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG controller is operating in host or device mode.

74.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. The port is now considered powered however this does not actually drive V_{BUS} on the USB, so a further action must be taken to control the external circuitry in the system so as to enable V_{BUS} generation.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

74.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.
3. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
4. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
5. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
6. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

74.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

74.15.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   – Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   – Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).
7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).
8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG host checks for space to write the disable

request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer
3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints. The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO).This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

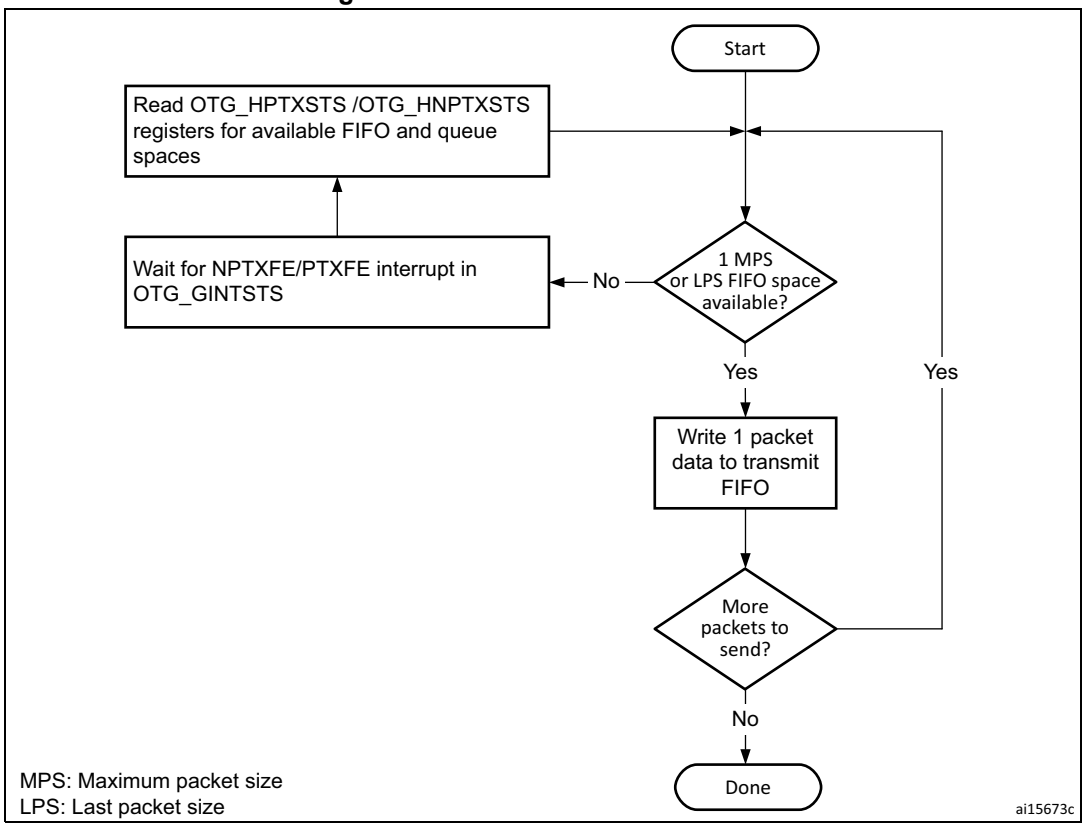
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG host determines the actual packet size based on the programmed maximum packet size and transfer size.

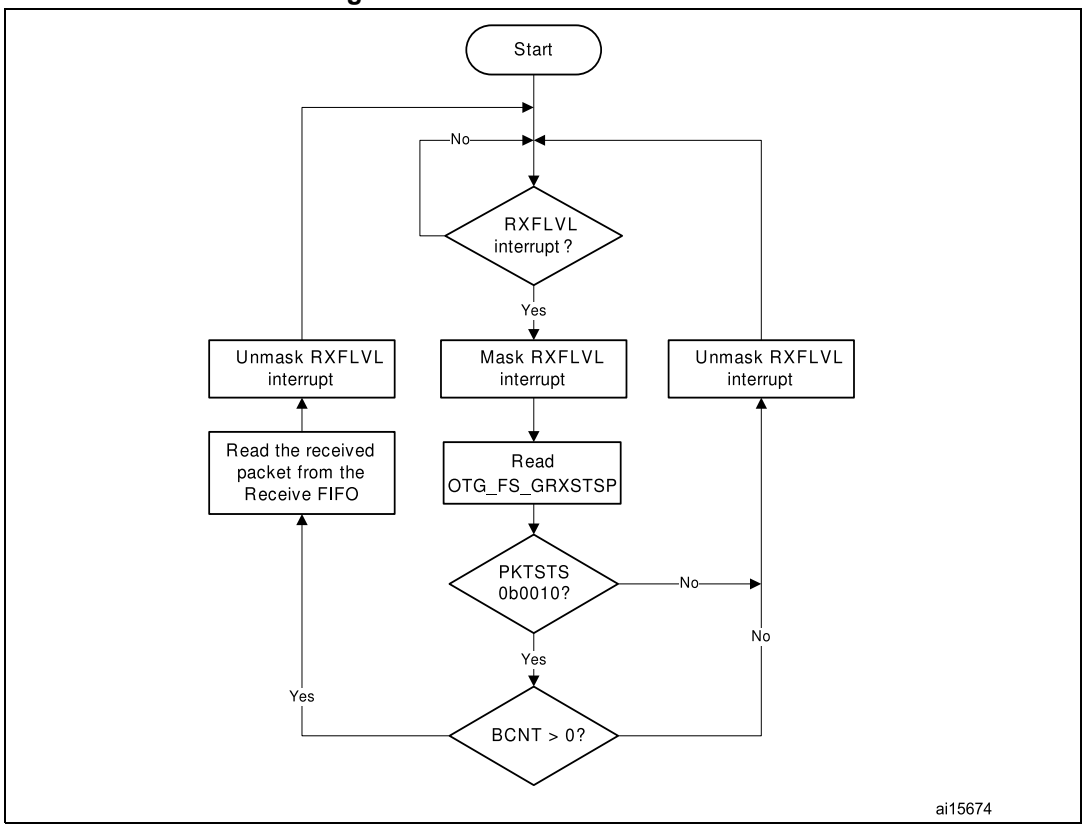
Figure 931. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 932. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 933*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

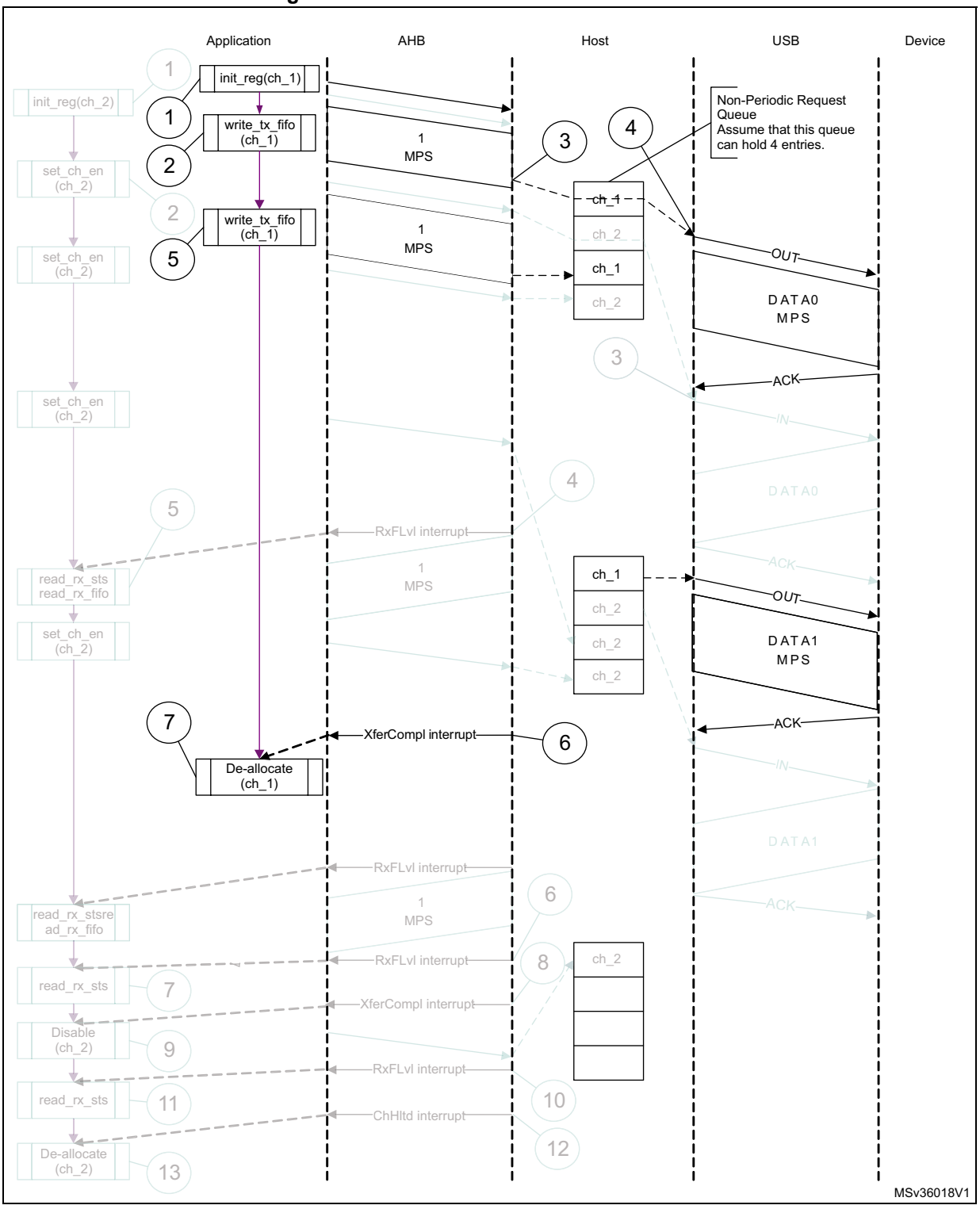
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  – The non-periodic transmit FIFO can hold two packets (1 Kbyte for HS).
  – The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 933. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

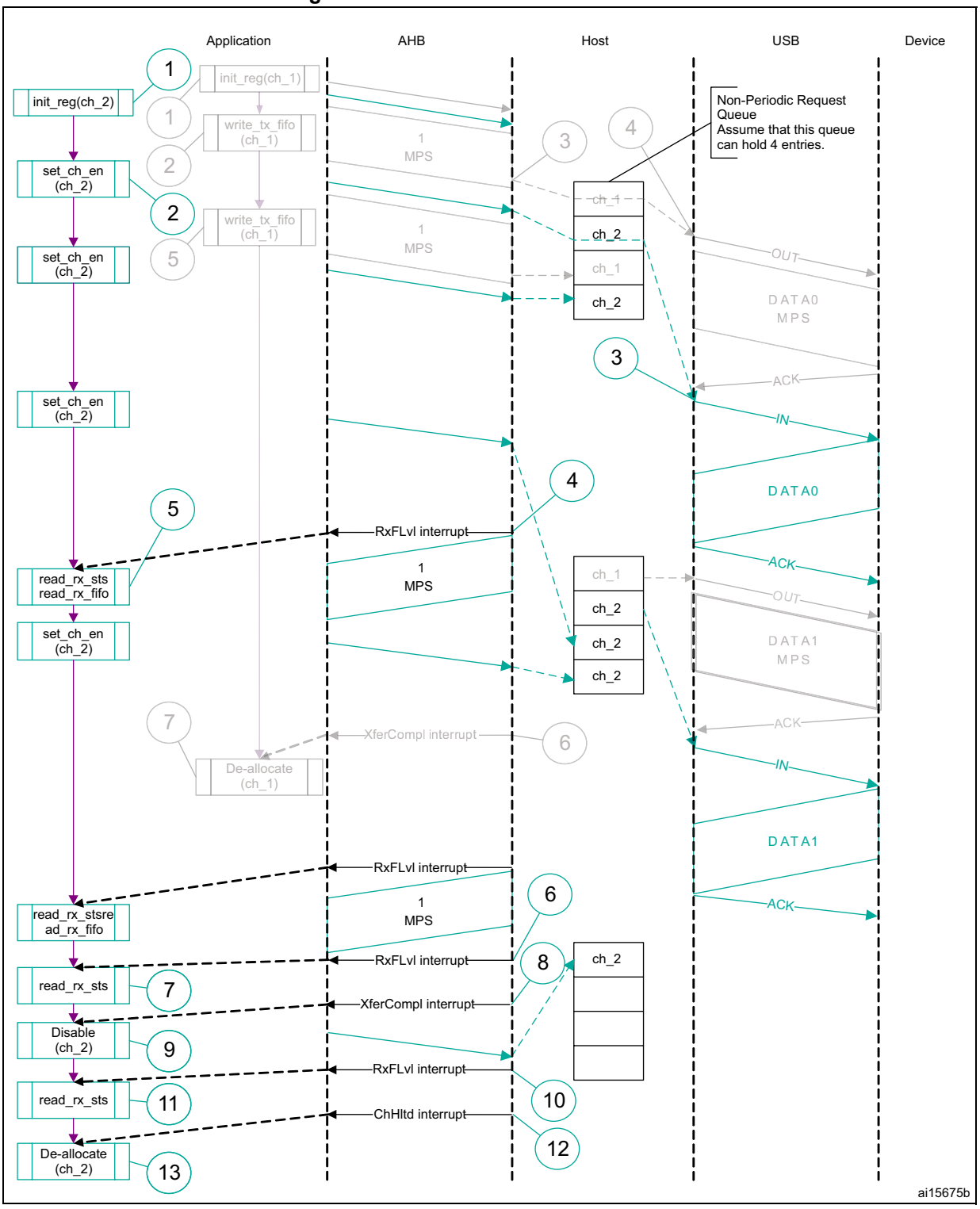
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 934*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).

  – The non-periodic request queue depth = 4.

Figure 934. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in

OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 935*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

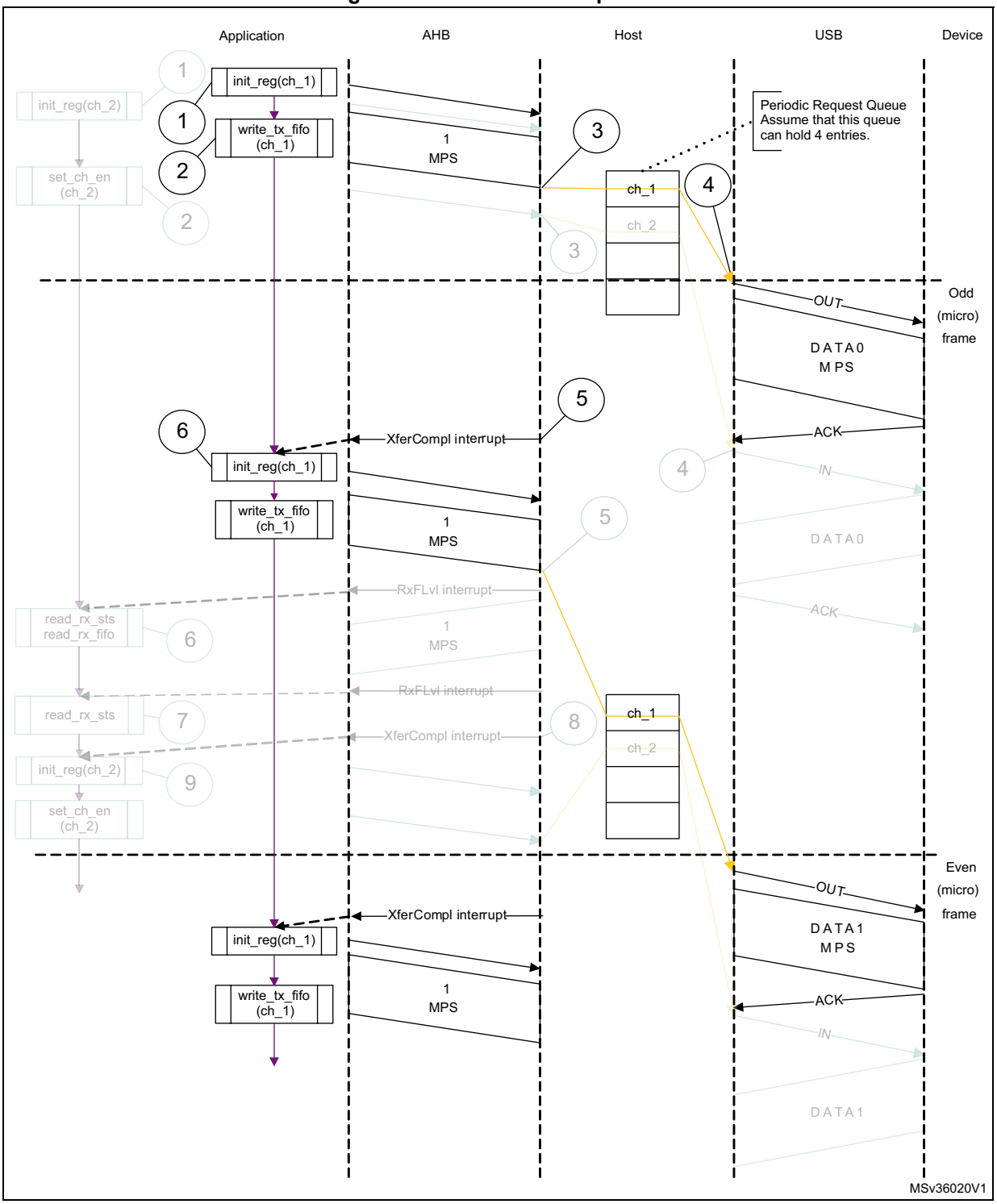
  – The periodic transmit FIFO can hold one packet (1 Kbyte)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send an OUT token in the next (odd) frame.
5. The OTG host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 935. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

`Unmask (NAK/TXERR/STALL/XFRC/FRMOR)`

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

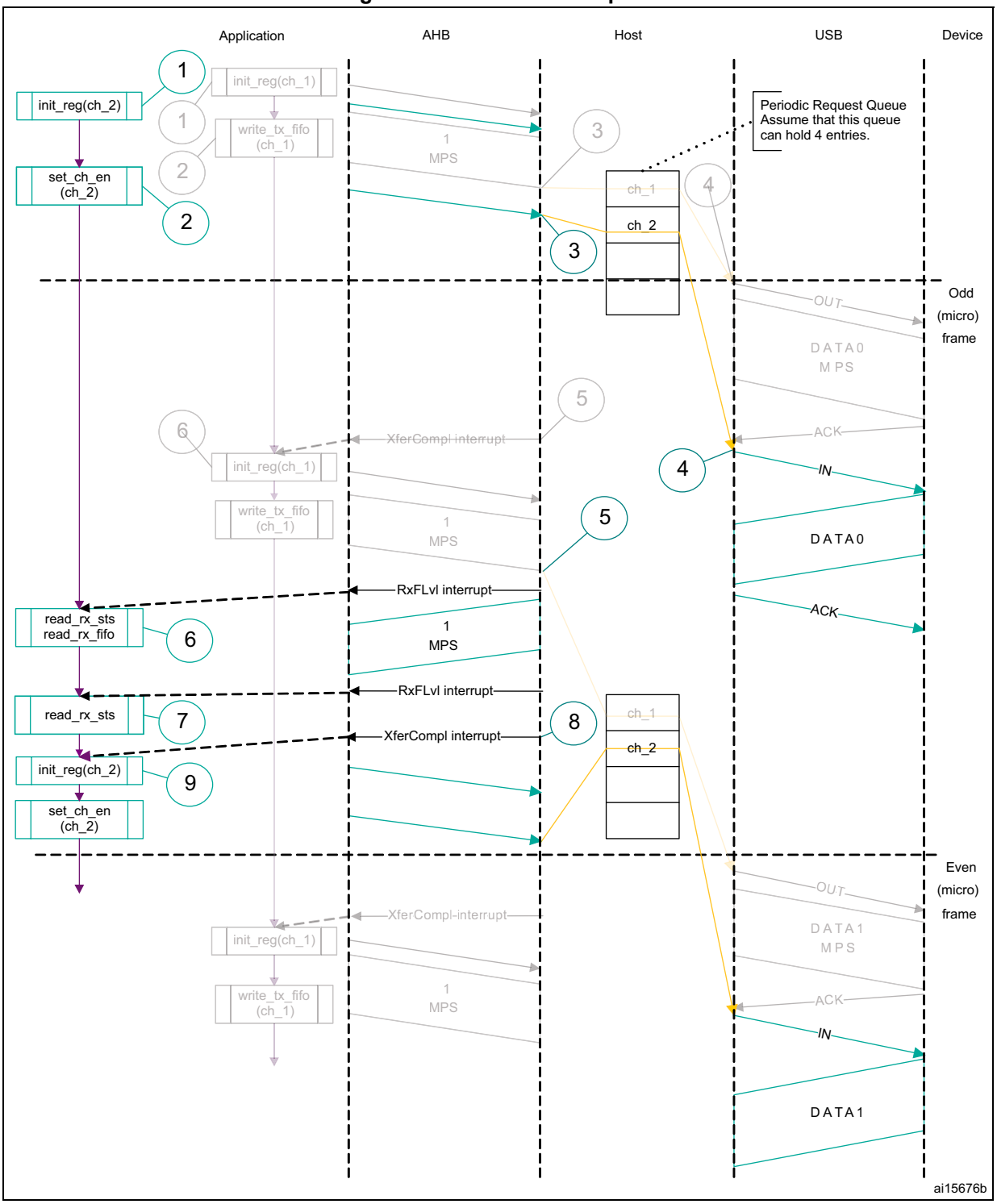
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 936. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 937*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum

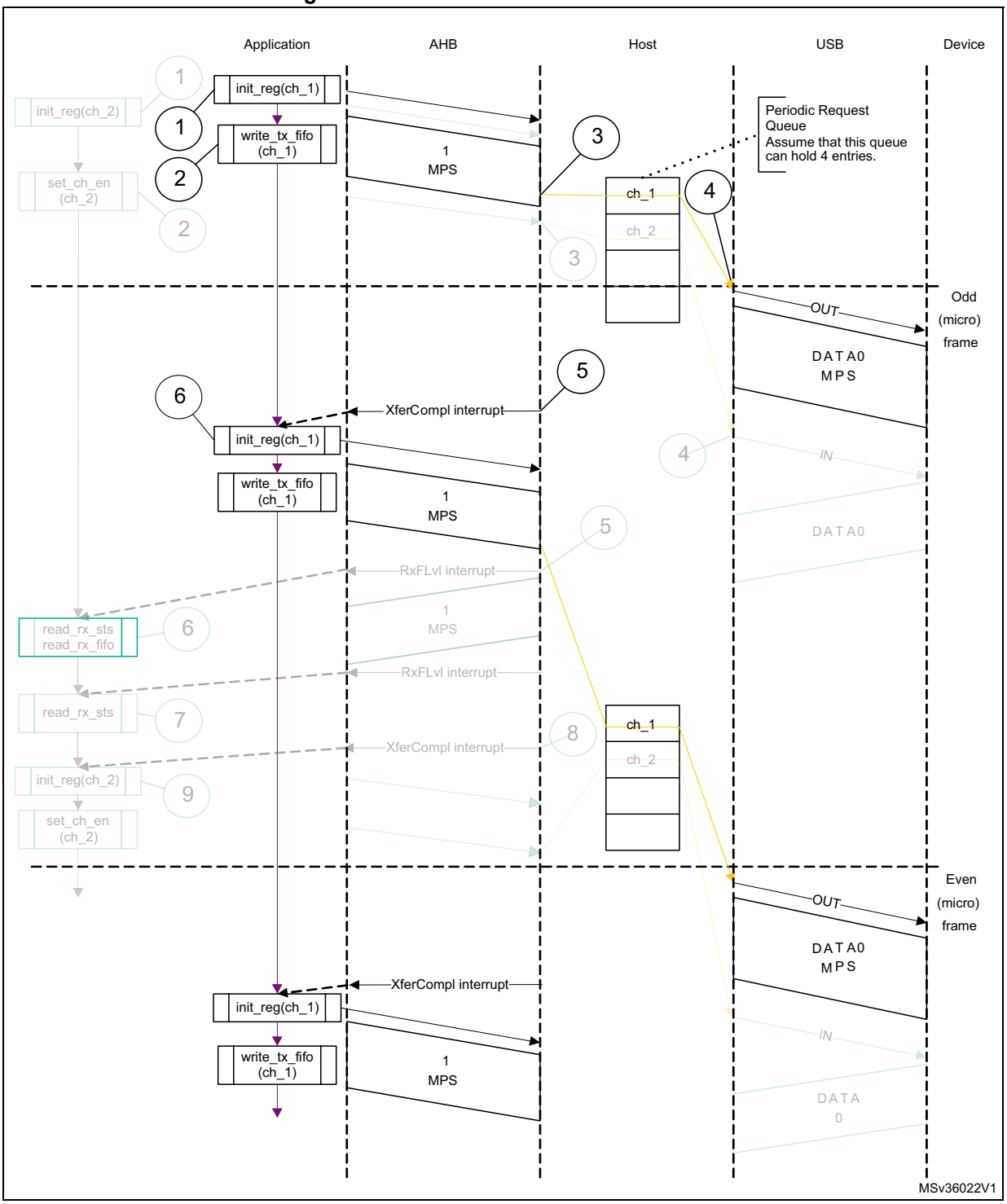
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send the OUT token in the next frame (odd).
5. The OTG host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 937. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
```

```
else
   {
   Re-initialize Channel
   }
}
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

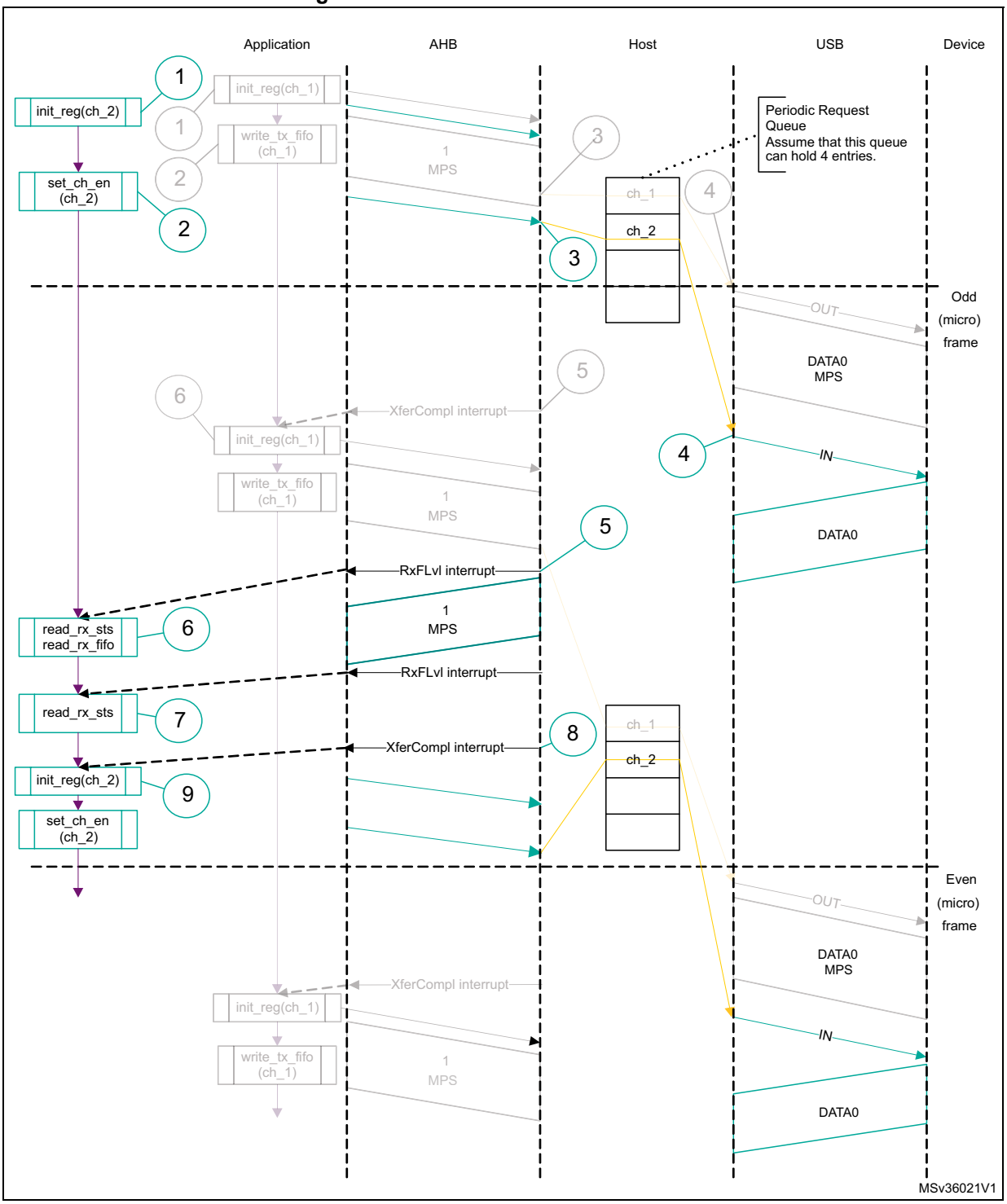
  – The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 938. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

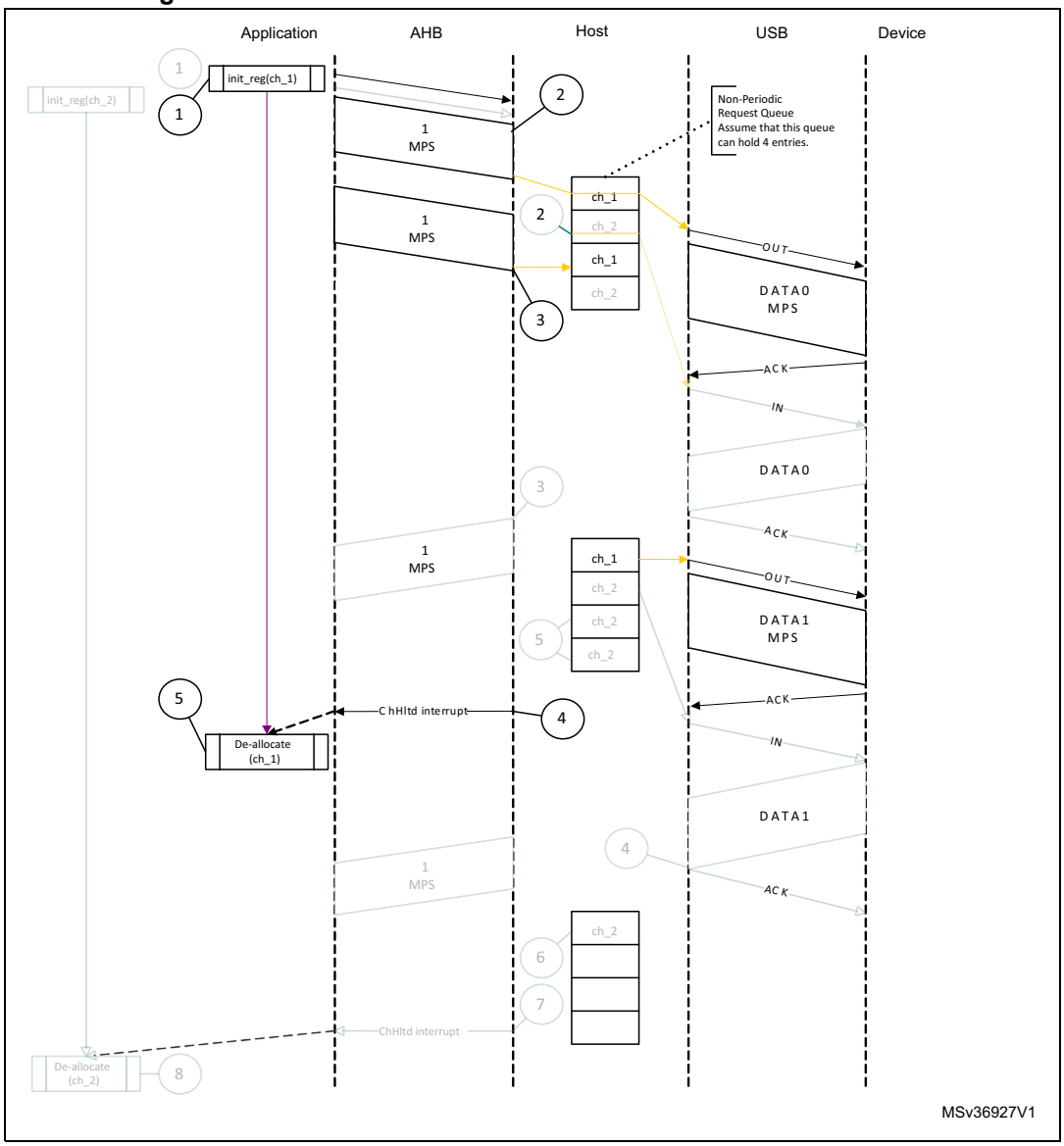
  When OTG controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable channel 1 as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG host uses the programmed DMA address to fetch the packet.
3. After fetching the last 32-bit word of the second (last) packet, the OTG host masks channel 1 internally for further arbitration.
4. The OTG host generates a CHH interrupt as soon as the last packet is sent.
5. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 939. Normal bulk/control OUT/SETUP transactions - DMA



- **NAK and NYET handling with internal DMA:**

1. The OTG host sends a bulk OUT transaction.
2. The device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.

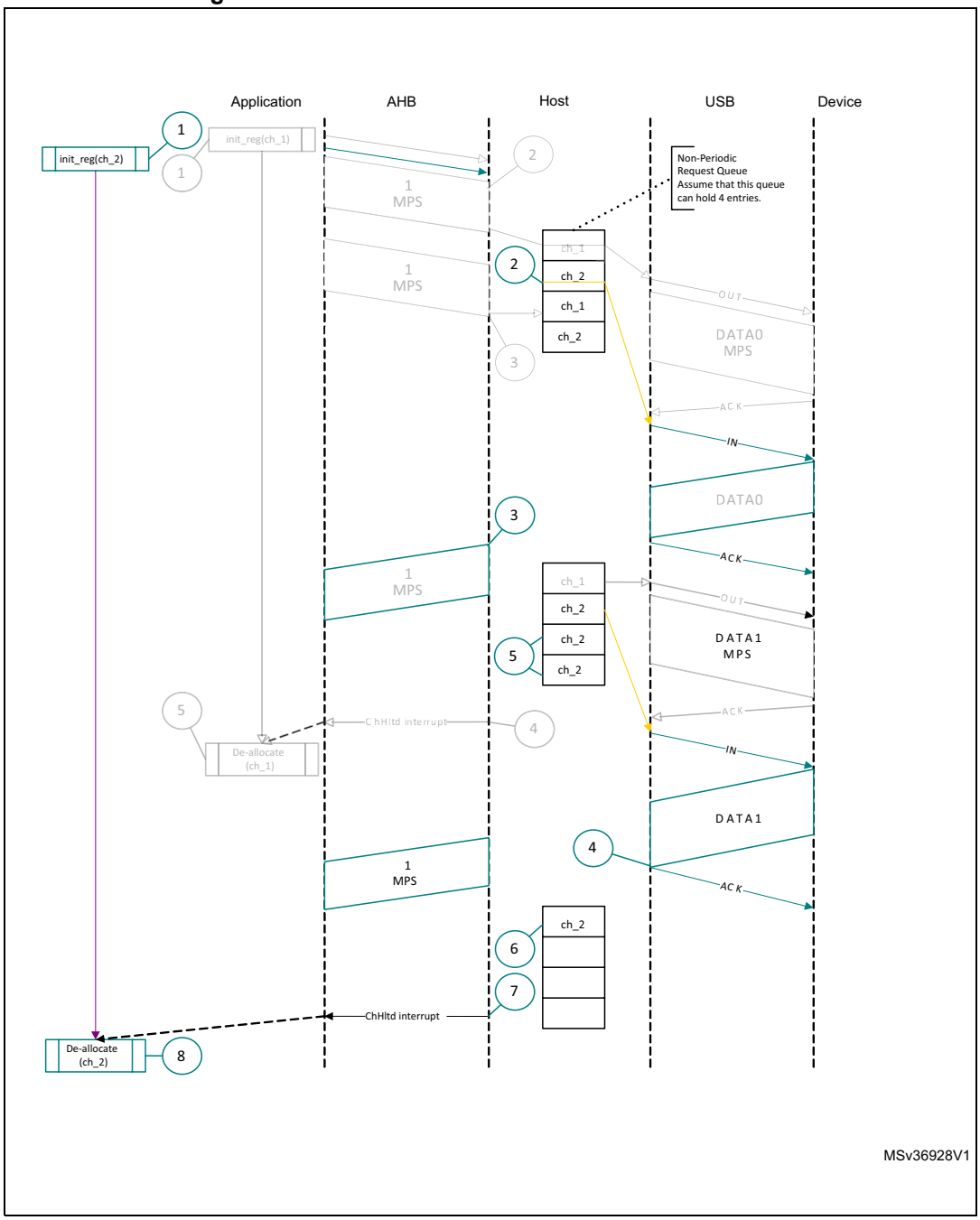
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
3. The OTG host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the OTG host sets an internal flag to remove any extra IN requests from the request queue.
5. The OTG host flushes the extra requests.
6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
7. The OTG host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
8. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 940. Normal bulk/control IN transaction - DMA

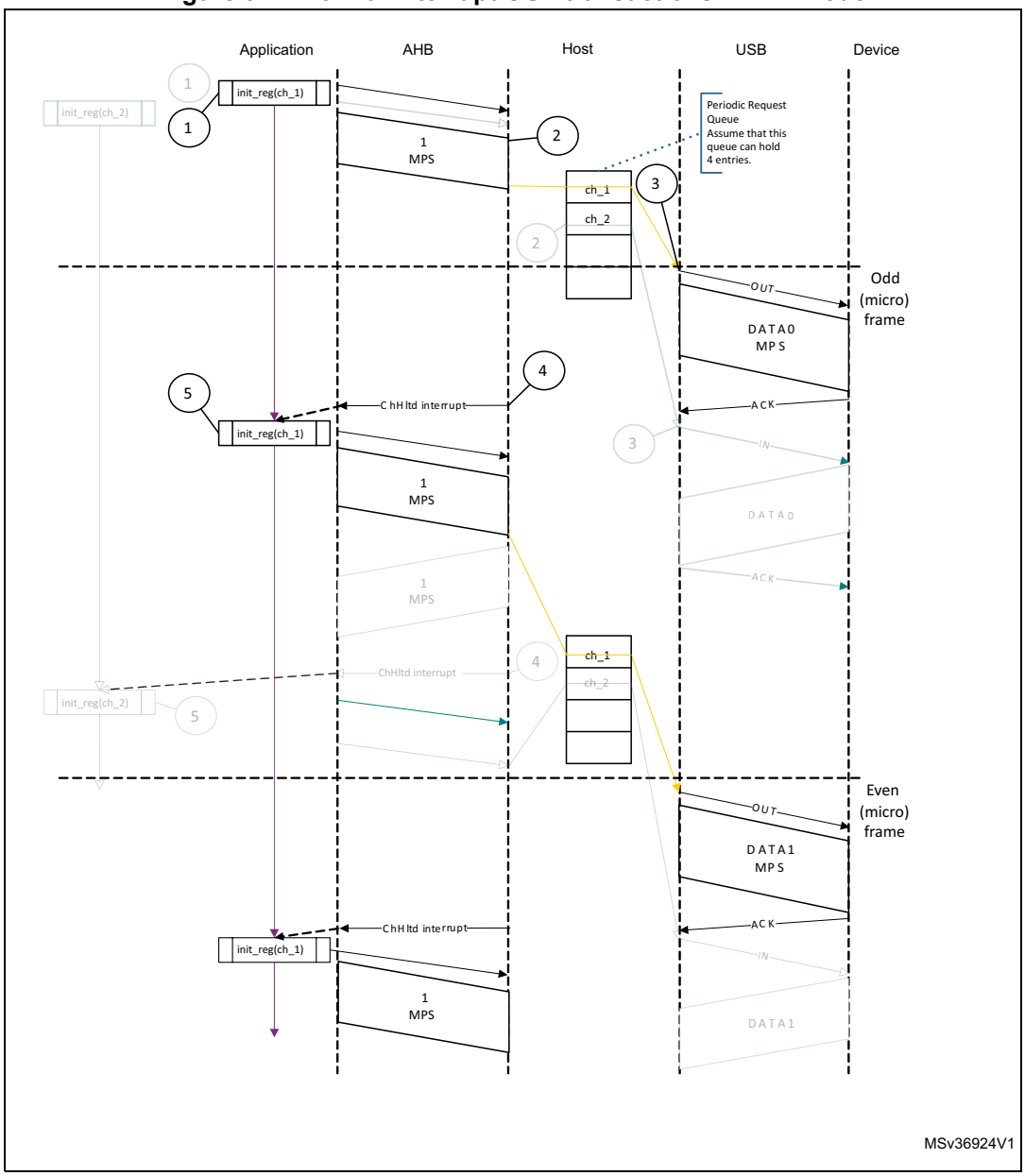


- **Interrupt OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.

4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 941. Normal interrupt OUT transactions - DMA mode



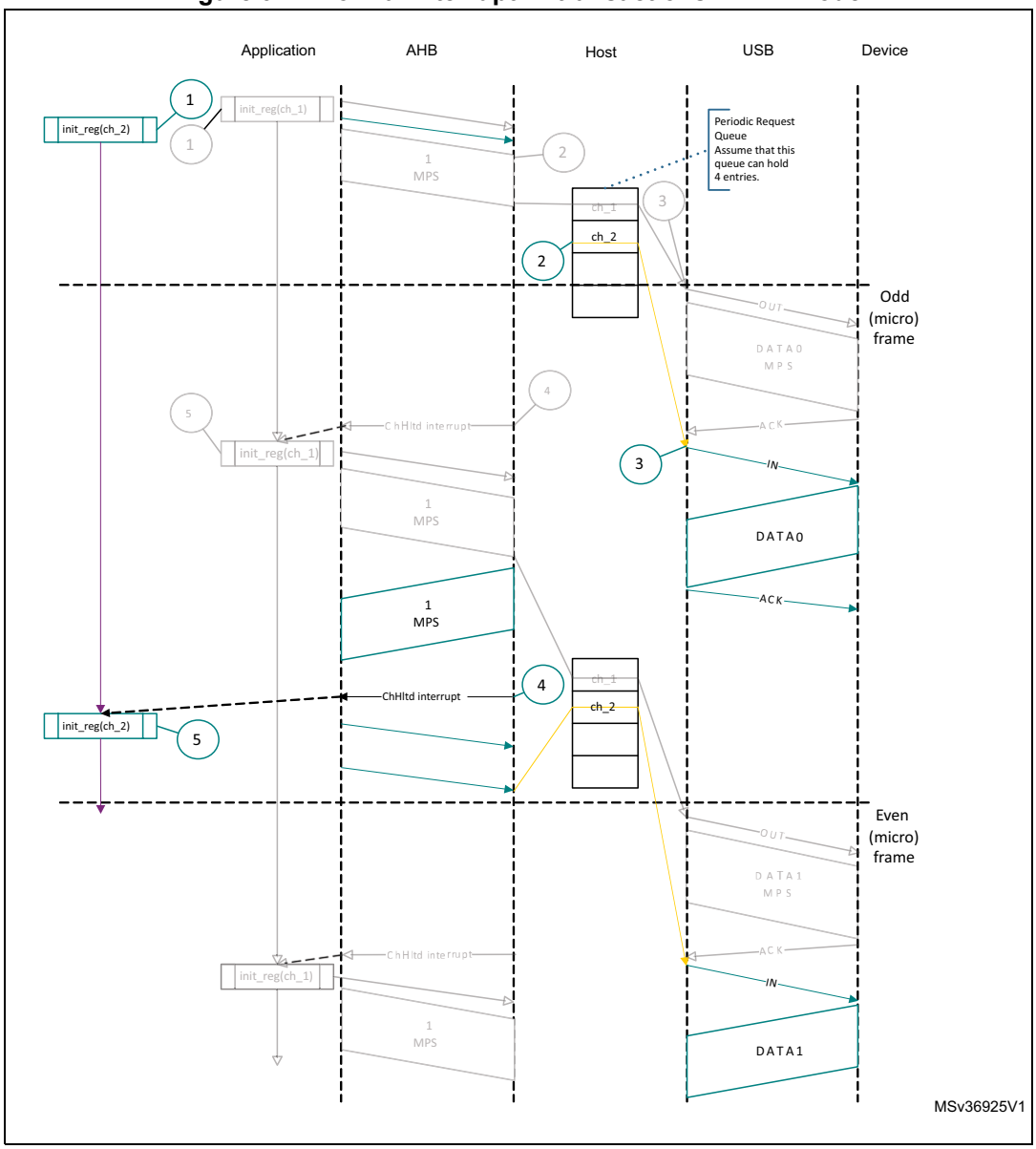
- **Interrupt IN transactions in DMA mode**

  The sequence of operations (channelx) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host writes consecutive writes up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 942. Normal interrupt IN transactions - DMA mode



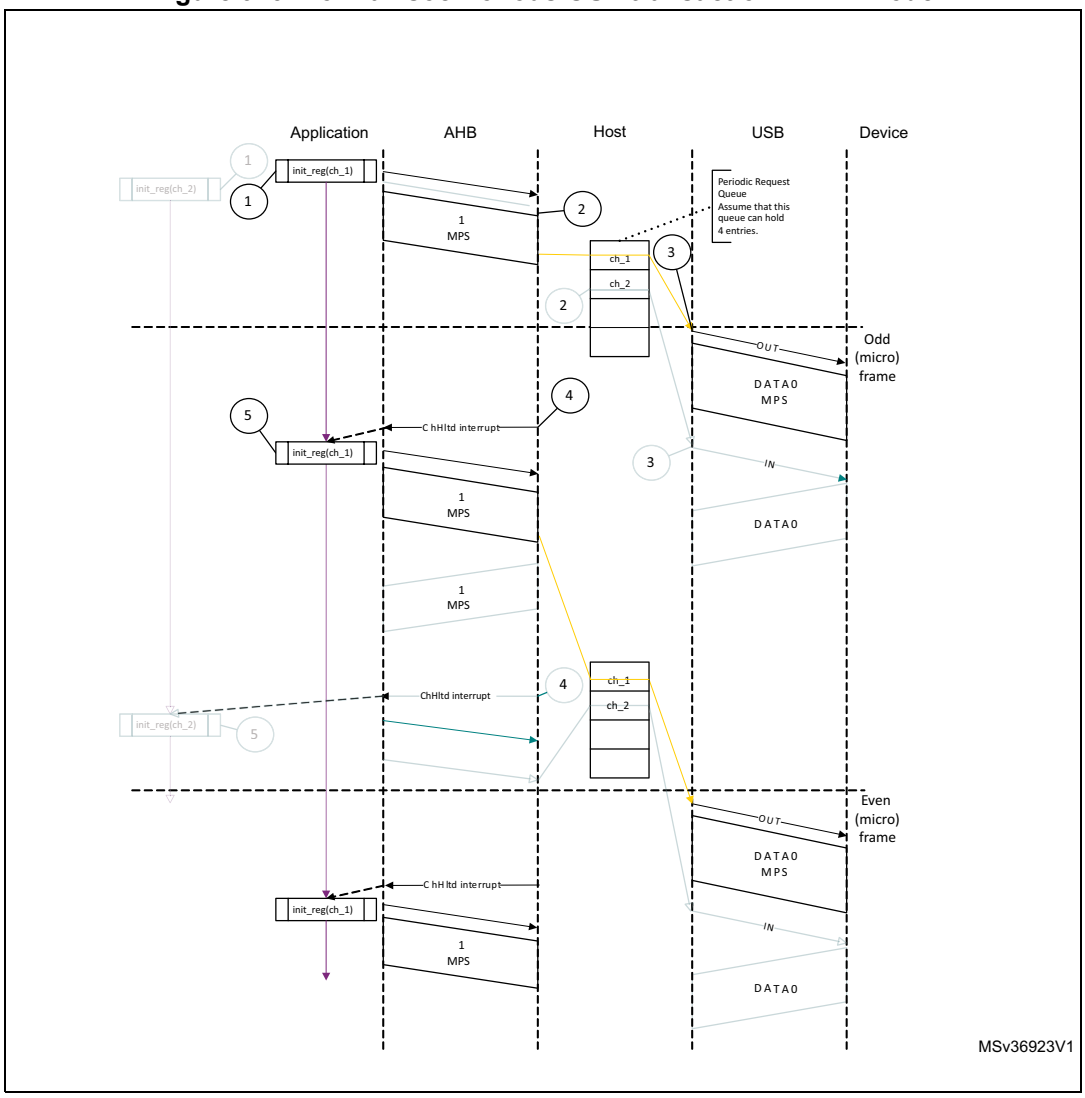
- **Isochronous OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth

transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.
4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 943. Normal isochronous OUT transaction - DMA mode



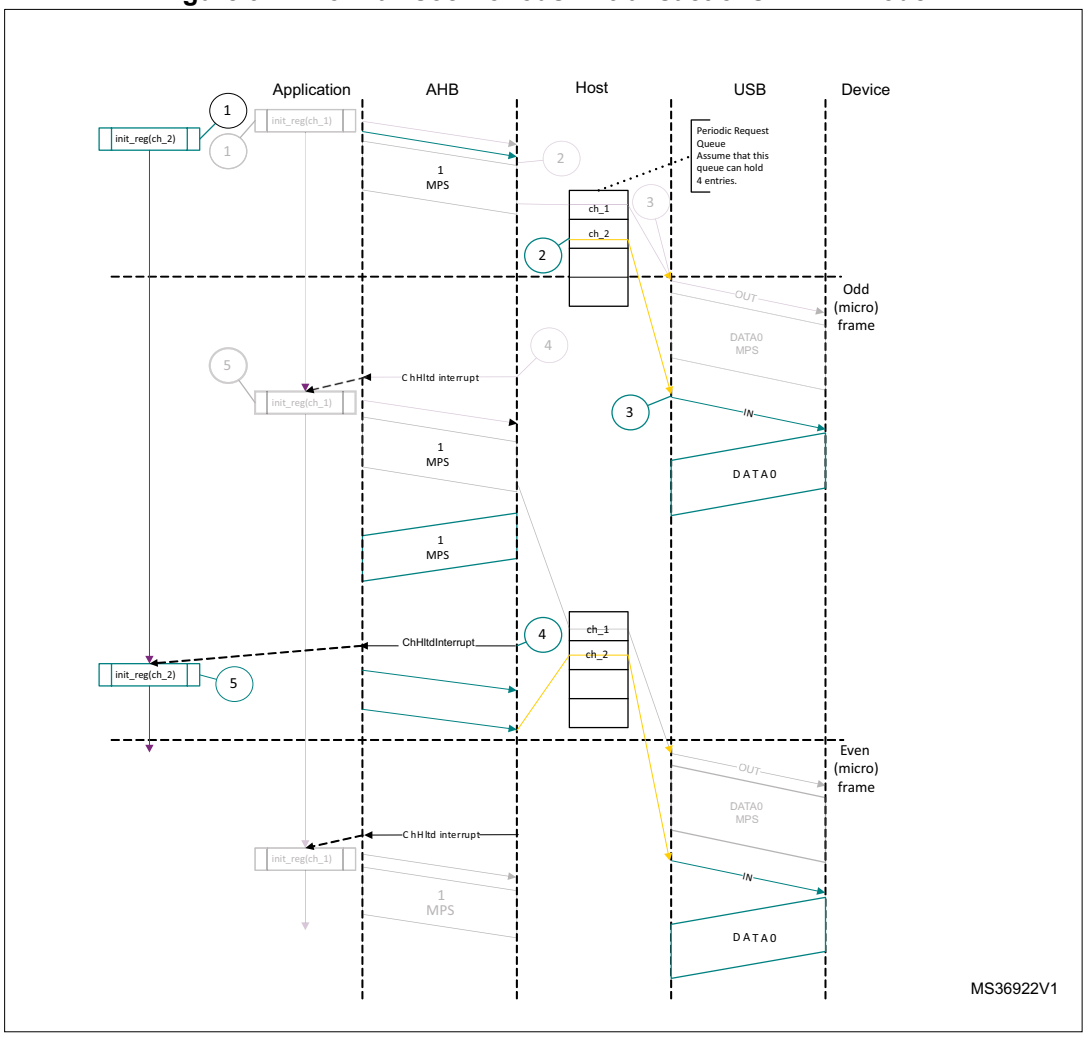
- **Isochronous IN transactions in DMA mode**

The sequence of operations ((channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host performs consecutive write operations up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 944. Normal isochronous IN transactions - DMA mode



- **Bulk and control OUT/SETUP split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.
3. After successfully transmitting start split, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
5. After successfully transmitting complete split, the OTG host generates the CHH interrupt.

6. In response to the CHH interrupt, de-allocate the channel.

- **Bulk/control IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG host masks the channel x internally for the arbitration after writing the request.
3. As soon as the IN token is transmitted, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.
5. The OTG host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
6. The OTG host starts writing the packet to the system memory after receiving the packet successfully.
7. As soon as the received packet is written to the system memory, the OTG host generates a CHH interrupt.
8. In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel 1 for start split as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. The OTG host starts reading the packet.
3. The OTG host attempts to send the start split transaction.
4. After successfully transmitting the start split, the OTG host generates the CHH interrupt.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
6. After successfully completing the complete split transaction, the OTG host generates the CHH interrupt.
7. In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.

6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.
7. The OTG host generates the CHH interrupt after transferring the received data to the system memory.
8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.
2. The OTG host starts reading the packet.
3. After successfully transmitting the start split (begin), the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).
5. After successfully transmitting the start split (end), the OTG host generates a CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.

   The OTG host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.

74.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   – SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   – INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   – OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   – STUPM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DIEPMSK
   – TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   – Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   – Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   – STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG in DMA mode, the OTG_DOEPDMA0 register must have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

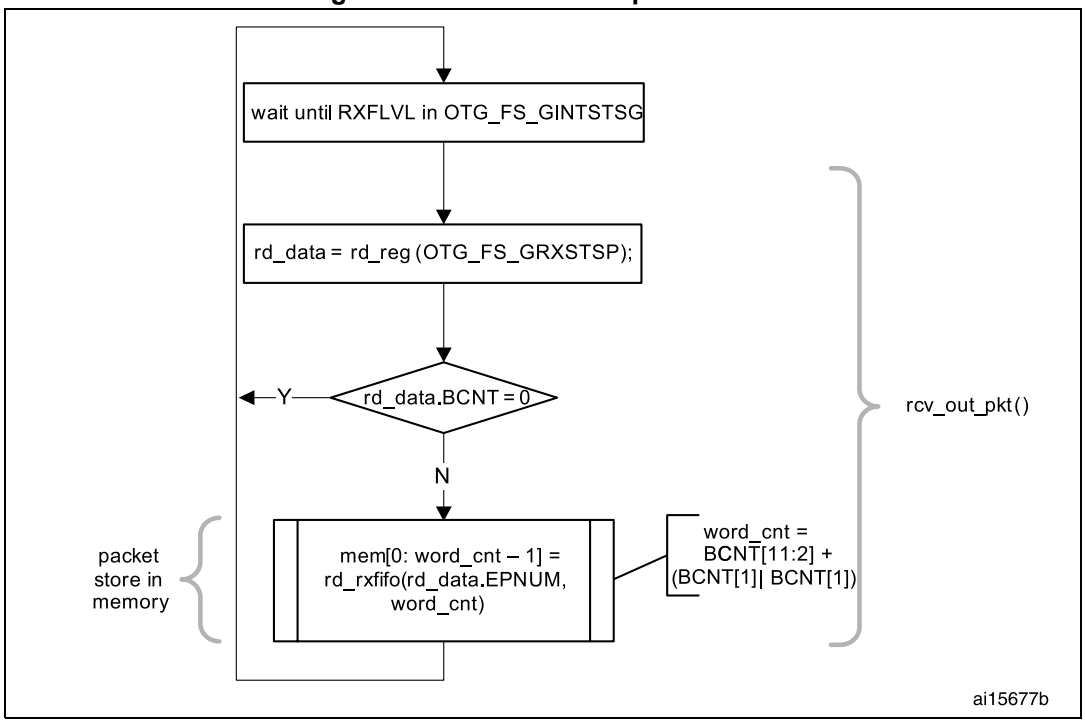
- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
   DPID = (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,
   DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
   PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num,
   DPID = (0b00).
   These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
   PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
   PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
   These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.
5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 945 provides a flowchart of the above procedure.

Figure 945. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   – 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the

receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

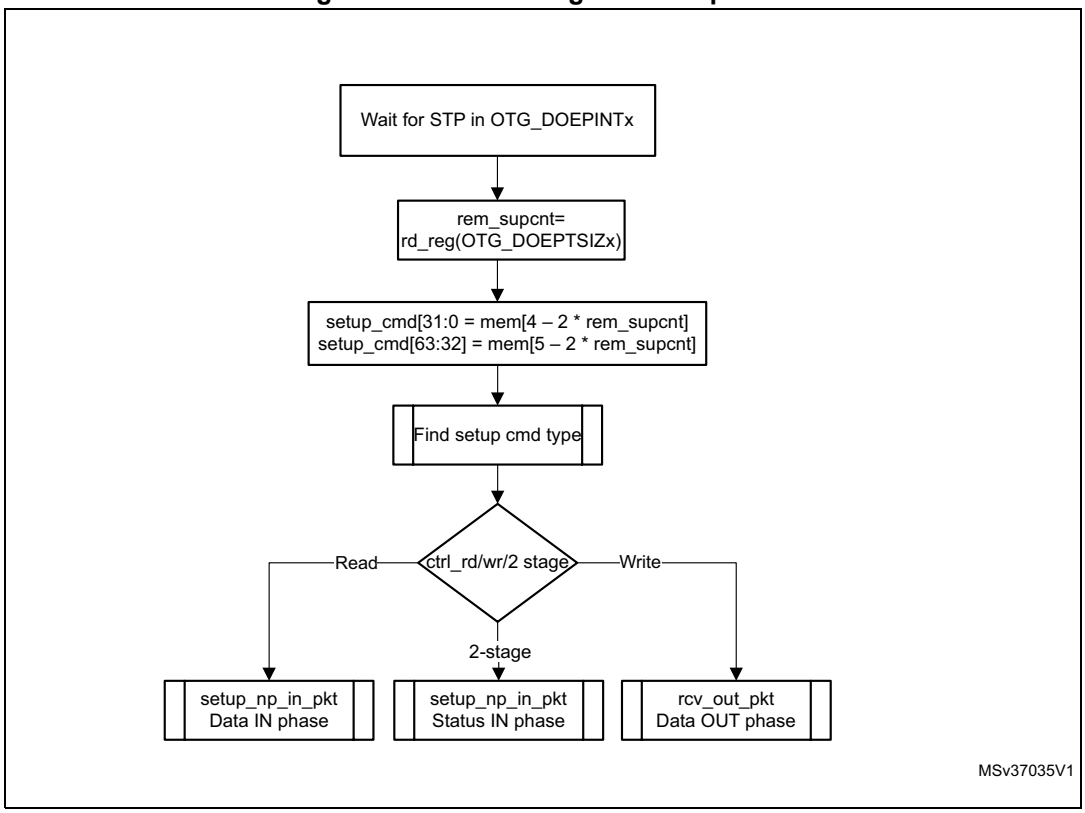
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 946. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Transfer Stop Programming for OUT endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_GRSTCTL.RXFFLSH =1
   – Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “*Setting the global OUT NAK*”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_DCTL
4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_DOEPCTLx
   – SNAK = 1 in registers OTG_DOEPCTLx
6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:
   – EPENA = 0 in registers OTG_DOEPCTLx
   – EPDIS = 0 in registers OTG_DOEPCTLx
   – SNAK = 0 in registers OTG_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint’s transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = n
   – $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint’s transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_DOEPCTLx
   – CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

     RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3The number of USB packets in which this payload was received =
     Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application may fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   – When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   – When the isochronous OUT data packet is received with CRC errors
   – When the isochronous OUT token received by the core is corrupted
   – When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At

this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   – EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

   The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

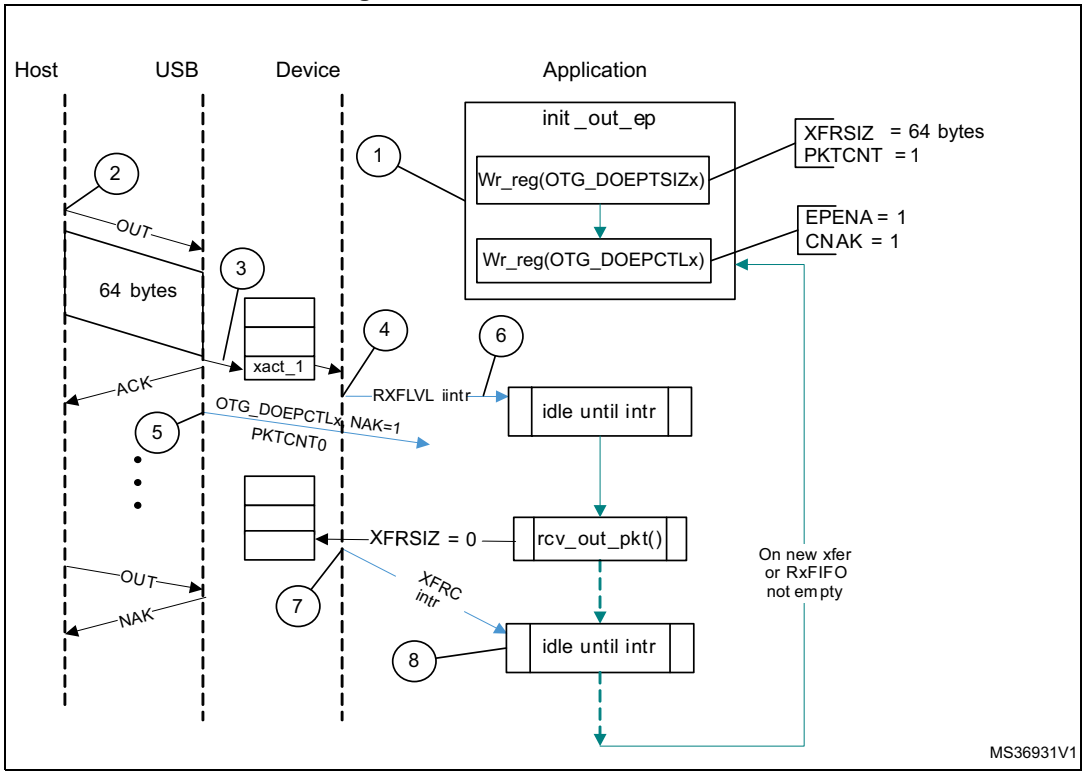
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 947 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 947. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   – In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   – To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).

– CNAK = 1 in OTG_DIEPCTLx

6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by setting:
   – EPDIS = 1 in all OTG_DIEPCTLx registers
2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   – EPDIS = 0 in OTG_DIEPCTLx
   – EPENA = 0 in OTG_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   – TXFFLSH = 1 in OTG_GRSTCTL
   – TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     Transfer size[EPNUM] = *x* × MPSIZ[EPNUM] + sp
     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = *x*
   – To transmit a single zero-length data packet:
     Transfer size[EPNUM] = 0
     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     First transfer: transfer size[EPNUM] = *x* × MPSIZ[epnum]; packet count = *n*;
     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
(where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = *x* + 1
Otherwise, packet count[EPNUM] = *x*;

MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO

mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it

eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_DIEPCTLx
   – EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

74.15.7 Worst case response time

When the OTG controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token may come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

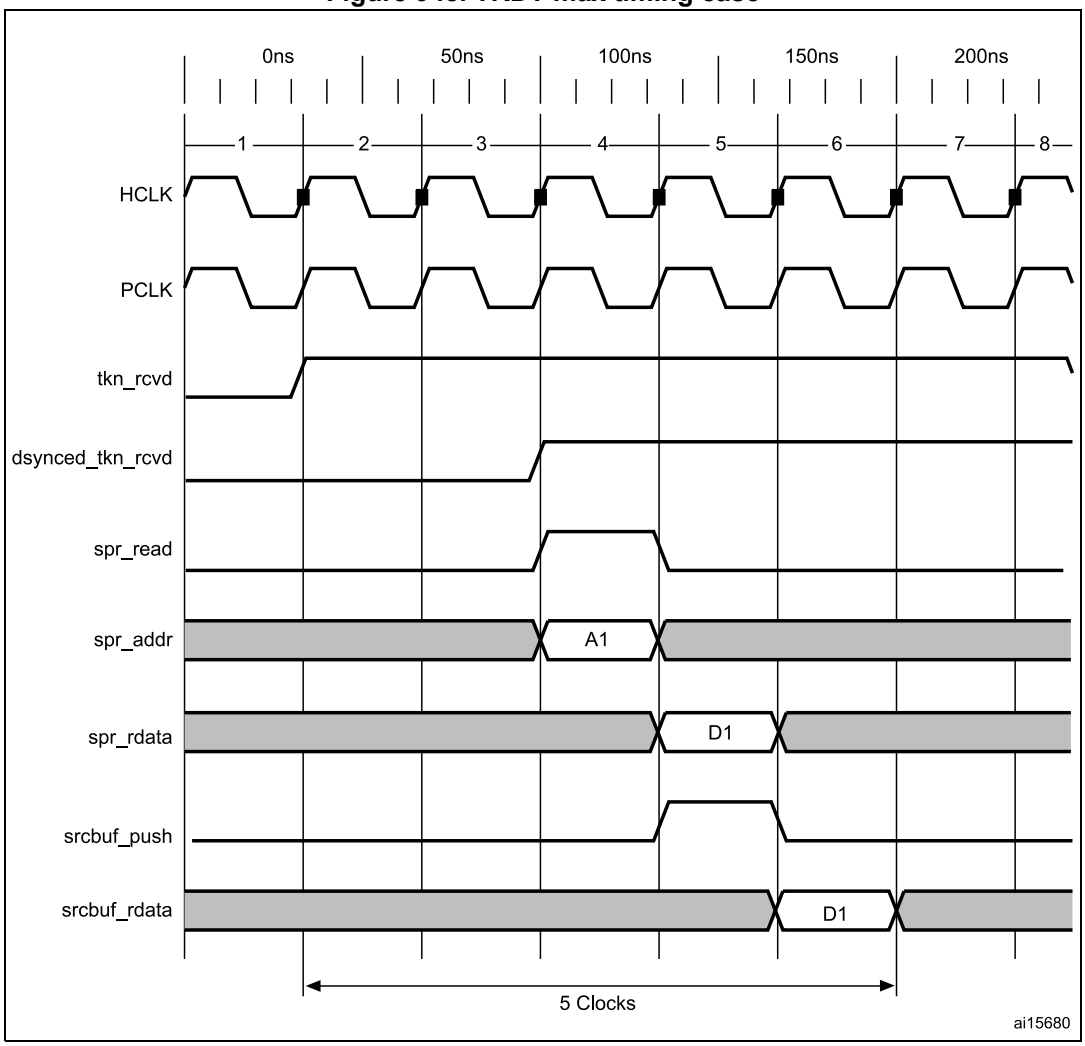
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 948 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 791: TRDT values*.

Figure 948. TRDT max timing case



74.15.8 OTG programming model

The OTG controller is an OTG device. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device.

75 USB2 EHCI/OHCI high/full/low-speed host controller (USBH)

75.1 Introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The USBH is a USB2 host controller supporting the standard registers used for full/low-speed (OHCI controller) and high-speed (EHCI controller).

75.2 USBH main features

- Connected to an internal PHY supporting high, full and low speeds.
- Supports OHCI model for full- and low-speed operation
- Supports EHCI model for high-speed operation
- LPM (link power management) support
- DMA engine with local 512 Byte packet buffer

The standards supported are:

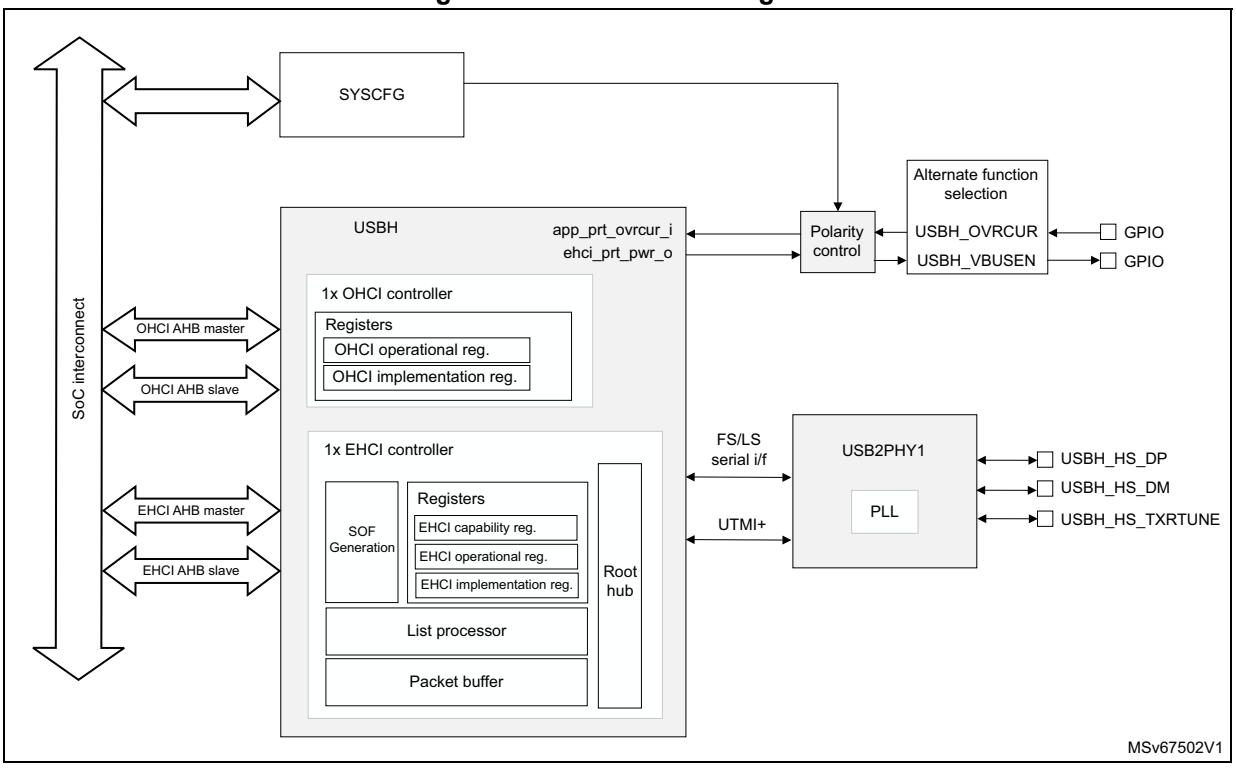
- Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 1.0, Intel Corp., March 12, 2002
  - http://www.intel.com
- EHCI 1.1 Addendum, Version 1.1, Intel Corp., August 2008
  - http://www.intel.com
- Open Host Controller Interface Specification for USB, Release 1.0a, Compaq, Inc., Microsoft Corp., National Semiconductor Corp., September 14, 1999
  - http://www.usb.org
- Universal Serial Bus Specification, Revision 2.0, USB Implementers Forum, Inc., April 27, 2000
  - http://www.usb.org
- USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Revision 1.05, Intel Corp., March 29, 2001
  - http://www.intel.com
- UTMI+ Specification, Revision 1.0, Philips Semiconductor, February 25, 2004
  - http://www.nxp.com

75.3 USBH functional description

75.3.1 USBH block diagram

The USBH block diagram is shown below:

Figure 949. USBH block diagram



75.3.2 Description of USBH packet buffer

The packet buffer provides storage and control for IN/OUT data. During OUT transactions, the system memory controller (inside the root hub) fetches data from the system memory and writes it here. During IN transactions, the root hub block writes the data. The packet buffer has a maximum depth of 512 bytes.

The packet buffer depth can be re-programmed during runtime. The depth selection influences the system latency and bandwidth allocated to the EHCI host controller. For example, if the packet buffer depth is set to the maximum possible value of 512 bytes, then for an IN transfer with USB packet size of 1024 bytes, there are two (1024/512) data transfers on the AHB bus.

75.3.3 Description of USBH root hub

The USBH root hub propagates the reset and resume signals to downstream ports and handles port connections and disconnections. In addition, the root hub is implemented with port router functionality to route the ports to either the EHCI host controller or OHCI host controller. The root hub operates on the local PHY clock, which operates on a free-running 60 MHz clock, and the clock source from each physical port, which operate at 60 MHz with an 8-bit interface.

75.4 USBH interrupts

Independent interrupts are generated from each controller:

- OHCI: usbh_ohci_it
  - HCI bus general interrupt (covers all 9 interrupt conditions covered by standard OHCI register HcInterruptStatus)
- EHCI: usbh_ehci_it
  - Interrupt async address
  - Host system error
  - Frame list rollover
  - Port change
  - USB error
  - USB interrupt

75.5 USBH registers

Note that USBH has two physical AHB slaves:

- USBH_OHCI (mapped to USBH_OHCI address block)[a]
- USBH_EHCI (mapped to USBH_EHCI address block)[a]

USBH registers described here cover both slaves.

75.5.1 USBH OHCI standard registers

The OHCI core is implemented according to USB 1.1 specification, release 1.0a. The standard OHCI operational registers are listed hereafter and in *Table 794: USBH register map and reset values* however their detailed description has been omitted in this document. Refer to the standard documentation for the register description. In the following list of registers, the prefix represents the address offset with respect to USBH_OHCI base address.

- [0x0000]: USBH_OHCI_HCREVISION
- [0x0004]: USBH_OHCI_HCCONTROL
- [0x0008]: USBH_OHCI_HCCOMMANDSTATUS
- [0x000C]: USBH_OHCI_HCINTERRUPTSTATUS
- [0x0010]: USBH_OHCI_HCINTERRUPTENABLE
- [0x0014]: USBH_OHCI_HCINTERRUPTDISABLE
- [0x0018]: USBH_OHCI_HCHCCA
- [0x001C]: USBH_OHCI_HCPERIODCURRENTED
- [0x0020]: USBH_OHCI_HCCONTROLHEADED
- [0x0024]: USBH_OHCI_HCCONTROLCURRENTED
- [0x0028]: USBH_OHCI_HCBULKHEADED
- [0x002C]: USBH_OHCI_HCBULKCURRENTED
- [0x0030]: USBH_OHCI_HCDONEHEAD

a. Refer to *Section 2.4 on page 184* for the register boundary addresses of each block.

- [0x0034]: USBH_OHCI_HCFMINTERVAL
- [0x0038]: USBH_OHCI_HCFMREMAINING
- [0x003C]: USBH_OHCI_HCFMNUMBER
- [0x0040]: USBH_OHCI_PERIODICSTART
- [0x0044]: USBH_OHCI_HCLSTHRESHOLD
- [0x0048]: USBH_OHCI_HCRHDESCRIPTORA
- [0x004C]: USBH_OHCI_HCRHDESCRIPTORB
- [0x0050]: USBH_OHCI_HCRHSTATUS
- [0x0054]: USBH_OHCI_HCRHPORTSTATUS0

75.5.2 USBH OHCI implementation register #6 (USBH_OHCI_INSNREG06)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_ ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **AHB_ECAP**: AHB error captured

Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to this register.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:9 **HBURST[2:0]**: HBURST value of the control phase at which the AHB error occurred

Bits 8:4 **EXP_BEATS[4:0]:** Number of expected beats

Number of beats expected in the burst where the AHB error occurred. Valid values are 0 and 4.

0: When no error, 0 is written to EXP_BEATS[4:0].

4: When INCR4 and an error has occurred, 4 is written to EXP_BEATS[4:0].

Other values except 4 are not written to EXP_BEATS[4:0].

Bits 3:0 **SUC_BEATS[3:0]**: Number of successful beats

Number of successfully completed beats in the current burst before the AHB error has occurred.

75.5.3 USBH OHCI implementation register #7 (USBH_OHCI_INSNREG07)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_MST_ERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB_MST_ERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AHB_MST_ERR[31:0]**: AHB master error address

AHB address of the control phase at which the AHB error has occurred. This field is valid only when INSNREG06 [31] is set.

75.5.4 USBH EHCI standard registers

The EHCI core is implemented according USB 1.1 specification release 1.0a, and EHCI Addendum 1.1. The standard EHCI registers are listed hereafter and in *Table 794: USBH register map and reset values* however their detailed description has been omitted in this document. Refer to the standard documentation for the register description. In the following list of registers, the prefix represents the address offset with respect to USBH_EHCI base address.

- EHCI capability registers:
  - [0x0000]: USBH_EHCI_HCCAPBASE
  - [0x0004]: USBH_EHCI_HCSPARAMS
  - [0x0008]: USBH_EHCI_HCCPARAMS
- EHCI functional registers
  - [0x0010]: USBH_EHCI_USBCMD
  - [0x0014]: USBH_EHCI_USBSTS
  - [0x0018]: USBH_EHCI_USBINTR
  - [0x001C]: USBH_EHCI_FRINDEX
  - [0x0020]: USBH_EHCI_CTRLDSSEGMENT
  - [0x0024]: USBH_EHCI_PERIODICLISTBASE
  - [0x0028]: USBH_EHCI_ASYNCLISTADDR
  - [0x0050]: USBH_EHCI_CONFIGFLAG
  - [0x0054]: USBH_EHCI_PORTSC0

75.5.5 USBH EHCI implementation register #1 (USBH_EHCI_INSNREG01)

This register controls the programmable packet buffer OUT/IN thresholds. The value specified here is the number of 32-bit words.

The OUT threshold is used to start the USB transfer as soon as the OUT threshold amount of data is fetched from system memory. It is also used to disconnect the data fetch, if the required space to hold the threshold amount of data is not available in the packet buffer.

The IN threshold is used to start the memory transfer as soon as the IN threshold amount of data is available in the packet buffer. It is also used to disconnect the data write, if the threshold amount of data is not available in the packet buffer.

The minimum OUT and IN threshold amount that can be programmed through INSN registers is 16 bytes. The minimum threshold amount that can be programmed is the highest possible INCRX burst value, that is INCR16. The minimum OUT and IN threshold value should be 64 bytes (16×32-bit words).

OUT and IN threshold values can be equal to the packet buffer depth only when isochronous/interrupt transactions are not initiated by the host controller.

Note: The break memory transfer bit is always enabled.

Depending on packet buffer depth settings, the most significant bits may be unused.

Power on reset sets these registers to reset values, soft reset does not affect these registers.

Address offset: 0x0094

Reset value: 0x0020 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT_THRESHOLD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN_THRESHOLD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OUT_THRESHOLD[7:0]**: The value specified here is the number of 32-bit words

The OUT threshold is used to start the USB transfer as soon as the OUT_THRESHOLD amount of data is fetched from system memory. It is also used to disconnect the data fetch, if the required space to hold the threshold amount of data is not available in the packet buffer.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IN_THRESHOLD[7:0]**: The value specified here is the number of 32-bit words

The IN threshold is used to start the memory transfer as soon as the IN_THRESHOLD amount of data is available in the packet buffer. It is also used to disconnect the data write, if the threshold amount of data is not available in the packet buffer.

75.5.6 USBH EHCI implementation register #2 (USBH_EHCI_INSNREG02)

Programmable packet buffer depth.

Address offset: 0x0098

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT_BUF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PKT_BUF[7:0]**: Programmable packet buffer depth
Programmable packet buffer depth.
The value specified here represents the number of 32-bit words.
Maximum valid packet buffer depth is 128 (32-bit words).

75.5.7 USBH EHCI implementation register #3 (USBH_EHCI_INSNREG03)

Address offset: 0x009C

Reset value: 0x0001 2001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE_LS_GLITCH
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EN_CLK256_CHECK	TESTSE_NAK	TX_TRA_DELAY[2:0]			FRM_LST_FETCH	TIME_AVAIL_OFF[7:0]								BRK_MEM_TRANS
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ENABLE_LS_GLITCH**: Enable/disable enhancement for line state glitch

By default this enhancement is enabled.

0: disabled

1: enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **EN_CLK256_CHECK**: Enable 256 clock checking

This bit controls the end of resume sequence of the EHCI host controller. By default, the value of this bit is 0 and during the end of resume sequence, the host controller waits for SE0 on the linestate before switching the PHY to high-speed.

When set to 1, during the end of resume sequence, the controller waits for SE0 or 256 clocks before switching the PHY to high-speed.

Setting this bit to 1, enables the 256 clock checking. Some of the UTMI PHYs do not present SE0 on the linestate during the end of resume sequence. For such PHYs, this bit should be set, so that the core does not wait forever for SE0.

This bit should be set only during initialization.

Bit 13 **TESTSE_NAK**: TestSE NAK

Ignore linestate during TestSE0 Nak

When set to 1 (default), the core ignores the linestate checking when transmitting SOF during the SE0_NAK test mode.

When set to 0, the port state machine disables the port if it does not find the linestate to be in SE0 when transmitting SOF during the SE0_NAK testing. While doing impedance measurement during the SE0_NAK testing, the linestate can go to non SE0 forcing the core to disable the port. This bit is used to control the port behavior during this state.

Bits 12:10 **TX_TRA_DELAY[2:0]**: Tx-Tx turnaround delay add on

This field specifies the extra delays in phy_clks to be added to the "Transmit to Transmit turnaround delay" value maintained in the core. The default value of this register field is 0. This default value of 0 is sufficient for most PHYs. But for some PHYs that put wait states during the token packet, it may be required to program a value greater than 0 to meet the transmit to transmit minimum turnaround time. The recommendation is to use the default value of 0 and change it only if there is an issue with minimum transmit-to-transmit turnaround time. This value should be programmed during core initialization and should not be changed afterwards.

Bit 9 **FRM_LST_FETCH**:

Setting this bit forces the host controller to fetch the periodic frame list in every microframe of a frame. If not set, then the periodic frame list is fetched only in microframe 0 of every frame. The default is 0 (not set). This bit can be changed only during core initialization and should not be changed afterwards.

Bits 8:1 **TIME_AVAIL_OFF[7:0]**: Time available offset

This value indicates the additional number of bytes to be accommodated for the time-available calculation.

The USB traffic on the bus can be started only when sufficient time is available to complete the packet within the EOF1 point. Refer to the USB 2.0 specification for details of the EOF1 point.

This time-available calculation is done in the hardware, and can be further offset by programming a value in this location.

Note: Time-available calculation is added for future flexibility. The application is not required to program this field by default.

Bit 0 **BRK_MEM_TRANS**: Break memory transfer

Used in conjunction with EHCI_H_INSNREG01 to enable breaking memory transactions into chunks once the OUT/IN threshold value is reached. Value is fixed to 1.

75.5.8 USBH EHCI implementation register #4 (USBH_EHCI_INSNREG04)

Address offset: 0x00A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPEND_SIGNAL	Res.	ASYNC_PIPELINE_DIS	Res.	HCCPARAMS_WRT	HCSPARAMS_WRT
										rw		rw		rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SUSPEND_SIGNAL**: Suspend signal

0: By default, the automatic feature is enabled. The suspend signal is de-asserted (logic level 1) when run/stop is reset by software, but the HCH (hchalted) bit is not yet set.

1: Disables the automatic feature, which takes all ports out of suspend when software clears the run/stop bit. This is for backward compatibility.

For systems where the host is halted without waking up all ports out of suspend, the port can become stuck because the clock coming from the PHY is not running when the halt is programmed. To avoid this, the EHCI automatically pulls ports out of suspend when the host is halted by software. This bit is used to disable this automatic function.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ASYNC_PIPELINE_DIS**: Asynchronous pipeline disable

The default value is 0 (not set). If set to 1, disables the fetch control state machine to queue further transaction to the asynchronous queue head if the USB transaction is pending for the queue head.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HCCPARAMS_WRT**:
When this bit is 1, the USBH_EHCI_HCCPARAMS register bits 17, 15:4, and 2:0 become writable.

Bit 0 **HCSPARAMS_WRT**:
When this bit is 1, the USBH_EHCI_HCSPARAMS register becomes writable.

75.5.9 USBH EHCI implementation register #5 (USBH_EHCI_INSNREG05)

Address offset: 0x00A4

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSY	VPORT [3]
														r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPORT[2:0]			VCONTROL_LDM	VCONTROL[3:0]				VSTATUS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VBUSY**: UTMI VBUSY
VBusy (read only). Hardware indicator that a write to this register has occurred and the hardware is currently processing the operation defined by the data written. When processing is finished, this bit is cleared.

Bits 16:13 **VPORT[3:0]**: UTMI VPORT
VPort (read write). Valid values for write are only 1 and 2. No other values should be written to this field. Supposing an illegal value, say 4, is written, any subsequent writes to this register are ignored and the read value is always the initial illegal value written (in this case 4).

Bit 12 **VCONTROL_LDM:** UTMI VCONTROLLoad (vendor control load)
Refer to the UTMI specification for detailed usage.
0: Load
1: NOP

Bits 11:8 **VCONTROL[3:0]**: UTMI VCONTROL (vendor control)
Vendor defined 4-bit parallel input bus. Refer to the UTMI specification for detailed usage.

Bits 7:0 **VSTATUS[7:0]**: UTMI VStatus (vendor status)
Refer to the UTMI specification for detailed usage.

75.5.10 USBH EHCI implementation register #6 (USBH_EHCI_INSNREG06)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **AHB_ECAP**: AHB error captured

Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to this bit.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:9 **HBURST[2:0]**: HBURST value of the control phase at which the AHB error occurred

Bits 8:4 **EXP_BEATS[4:0]:** Number of expected beats

Number of beats expected in the burst at which the AHB error has occurred. Valid values are 0 to 16.

0: When no error, 0 is written to EXP_BEATS[4:0].

4: When INCR4 and error has occurred, 4 is written to EXP_BEATS[4:0]

8: When INCR8 and error has occurred, 8 is written to EXP_BEATS[4:0]

16: When INCR16 and error has occurred,16 is written to EXP_BEATS[4:0]

Other values except 4, 8, and 16 are not written to EXP_BEATS[4:0]

Bits 3:0 **SUC_BEATS[3:0]**: Number of successful beats

Number of successfully-completed beats in the current burst before the AHB error has occurred.

75.5.11 USBH EHCI implementation register #7 (USBH_EHCI_INSNREG07)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_MST_ERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB_MST_ERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AHB_MST_ERR[31:0]**: AHB master error address

AHB address of the control phase at which the AHB error has occurred.

75.5.12 USBH register map

Table 794. USBH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	USBH_OHCI_HCREVISION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV[7:0]							
	Reset value																									0	0	0	1	0	0	0	0
0x004	USBH_OHCI_HCCONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWE	RWC	IR	HCFS[1:0]		BLE	CLE	IE	PLE	CBSR[1:0]	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x008	USBH_OHCI_HCCOMMAND STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR	BLF	CLF	HCR
	Reset value															0	0													0	0	0	0
0x00C	USBH_OHCI_HC INTERRUPT STATUS	Res.	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value		0																								0	0	0	0	0	0	0
0x010	USBH_OHCI_HC INTERRUPT ENABLE	MIE	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value	0	0																								0	0	0	0	0	0	0
0x014	USBH_OHCI_HC INTERRUPT DISABLE	MIE	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value	0	0																								0	0	0	0	0	0	0
0x018	USBH_OHCI_HCHCCA	HCCA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x01C	USBH_OHCI_HCPERIOD CURRENTED	PCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x020	USBH_OHCI_HCCONTROL HEADED	CHED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x024	USBH_OHCI_HCCONTROL CURRENTED	CCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x028	USBH_OHCI_HCBULK HEADED	BHED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x02C	USBH_OHCI_HCBULK CURRENTED	BCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x030	USBH_OHCI_HCDONE HEAD	DH[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x034	USBH_OHCI_HCFM INTERVAL	Res.	FSMPS[14:0]															Res.	Res.	FI[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			1	0	1	1	1	0	1	1	0	1	1	1	1	1

Table 794. USBH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	USBH_OHCI_HCFM REMAINING	FRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FR[13:0]													
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	USBH_OHCI_HCFM NUMBER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	USBH_OHCI_PERIODIC START	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	USBH_OHCI_HCLS THRESHOLD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LST[13:0]													
	Reset value																			0	0	0	1	1	0	0	0	1	0	1	0	0	0
0x048	USBH_OHCI_HCRH DESCRIPTORA	POTPGT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCP	OCPM	DT	NPS	PSM	NDP[7:0]							
	Reset value	0	0	0	0	0	0	1	0												0	0	0	0	0	0	0	0	0	0	0	1	0
0x04C	USBH_OHCI_HCRH DESCRIPTORB	PPCM[15:0]																DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	USBH_OHCI_HCRH STATUS	CRWE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCIC	LPSC	DRWE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCI	LPS
	Reset value	0														0	0	0													0	0
0x0054	USBH_OHCI_HCRHPORT STATUS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRSC	OCIC	PSSC	PESC	CSC	Res.	Res.	Res.	Res.	Res.	Res.	LSDA	PPS	Res.	Res.	Res.	PRS	POCI	PSS	PES	CCS
	Reset value												0	0	0	0	0							0	0				0	0	0	0	0
0x0098	USBH_OHCI_INSNREG06	AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H BURST[2:0]			EXP_BEATS [4:0]					SUC_ BEATS [3:0]			
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x009C	USBH_OHCI_INSNREG07	AHB_MST_ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0000	USBH_EHCI_HCCAP BASE	HCIVERSION[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPLENGTH[7:0]							
	Reset value	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0									0	0	0	1	0	0	0	0
0x0004	USBH_EHCI_HCSPARAMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPN[3:0]				Res.	Res.	Res.	P_INDICATOR	N_CC[3:0]				N_PCC [3:0]				PRR	Res.	Res.	PPC	N_PCC [3:0]			
	Reset value									0	0	0	0				0	0	0	0	1	0	0	1	0	0			1	0	0	1	0
0x0008	USBH_EHCI_HCCPARAMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMC	Res.	EECP[7:0]								ISO[3:0]				Res.	ASPC	PFLF	SFAC
	Reset value															1		1	0	1	0	0	0	0	0	0	0	1	0		1	1	0
0x0010	USBH_EHCI_USBCMD	Res.	Res.	Res.	Res.	HIRD[3:0]				ITC[7:0]								Res.	Res.	Res.	Res.	ASPME	Res.	ASPMC [1:0]		LHCR	IAAD	ASE	PSE	FLS[1:0]		HCR	RS
	Reset value					0	0	0	0	0	0	0	0	1	0	0	0					1		1	1	0	0	0	0	0	0	0	0

Table 794. USBH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	USBH_EHCI_USBSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASS	PSS	REC	HCH	Res.	Res.	Res.	Res.	Res.	Res.	IAA	HSE	FLR	PCD	USBERRINT	USBINT
	Reset value																	0	0	0	1							0	0	0	0	0	0
0x0018	USBH_EHCI_USBINTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAAE	HSEE	FLRE	PCIE	USBERRINTE	USBINTE
	Reset value																											0	0	0	0	0	0
0x001C	USBH_EHCI_FRINDEX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRINDEX[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	USBH_EHCI_CTRLDSSEGMENT	SEG_4G_SELECTOR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	USBH_EHCI_PERIODICLISTBASE	PERIODICLISTBASE[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x0028	USBH_EHCI_ASYNCLIST ADDR	LPL[31:5]																										Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x0050	USBH_EHCI_CONFIGFLAG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF
	Reset value																																0
0x0054	USBH_EHCI_PORTSC0				DEVADDR[6:0]				SS[1:0]		WKOC_E	WKDSCNNT_E	WKCNNT_E	PTC[3:0]				PIC[1:0]		PO	PP	LS[1:0]		SUSP_L1	PR	SUSPEND	FPR	OCC	OCA	PEDC	PED	CSC	CCS
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	USBH_EHCI_INSNREG01	Res.	Res.	Res.	Res.	Res.	Res.	Res.		OUT_THRESHOLD [7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN_THRESHOLD [7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0x0098	USBH_EHCI_INSNREG02	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							PKT_BUF[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x009C	USBH_EHCI_INSNREG03	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE_LS_GLITCH	Res.	EN_CLK256_CHECK	TESTSE_NAK	TX_TRA_DELAY[2:0]			FRM_LST_FETCH	TIME_AVAIL_OFF[7:0]								BRK_MEM_TRANS
	Reset value																1		0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0x00A0	USBH_EHCI_INSNREG04	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPEND_SIGNAL	Res.	ASYNC_PIPELINE_DIS	Res.	HCCPARAMS_WRT	HCSPARAMS_WRT
	Reset value																											0		0		0	0

Table 794. USBH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00A4	USBH_EHCI_INSNREG05	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSY	VPORT[3:0]				VCONTROL_LDM	VCONTROL[3:0]				VSTATUS[7:0]							
	Reset value															0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	USBH_EHCI_INSNREG06	AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	USBH_EHCI_INSNREG07	AHB_MST_ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4* for the register boundary addresses.

76 Ethernet subsystem overview (ETHSS)

76.1 Introduction

This section describes the STM32MP21xx Ethernet subsystem.

The Ethernet subsystem includes two independent TSN Gigabit Ethernet controllers.

This section includes the subsystem overview, details the interconnections between the peripherals and provides the subsystem-related registers description.

For more information on the Ethernet controller, refer to *Section 77: Ethernet (ETH): gigabit media access control (GMAC) with DMA controller*

76.2 ETHSS main features

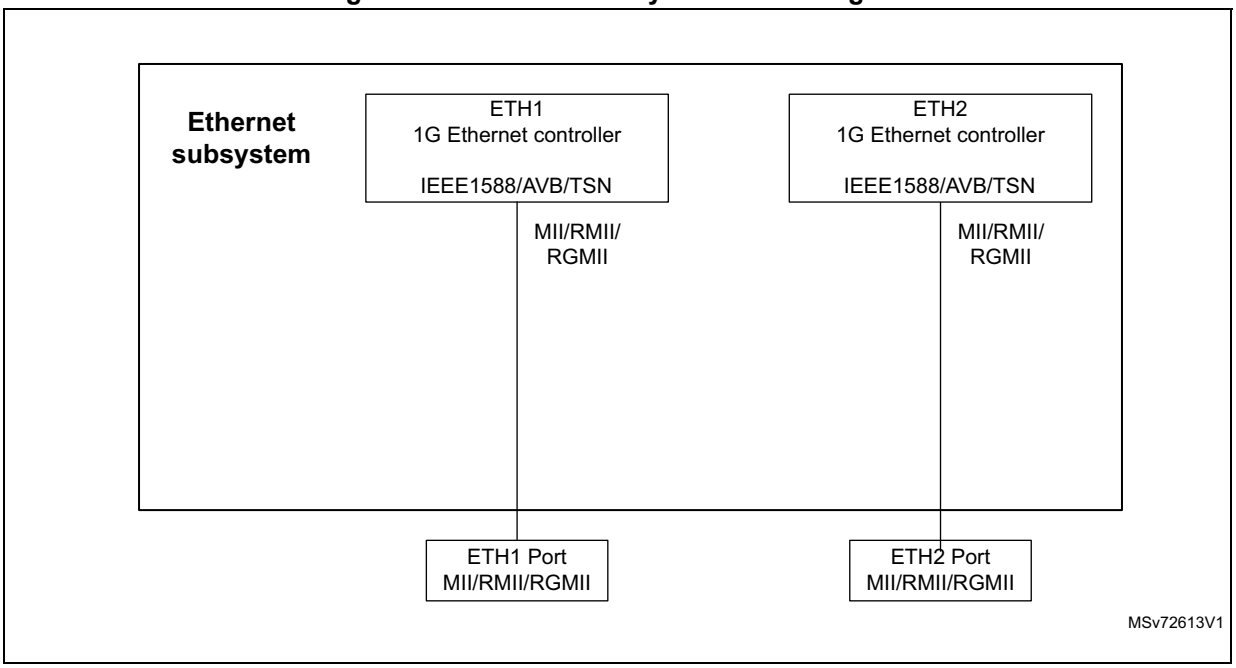
- Two fully independent instances of 10/100/1G Ethernet MAC controller, enabling transmission and reception of data over Ethernet, in compliance with IEEE 802.3-2008. Each Ethernet MAC controller is connected to an external Ethernet PHY via a standard media independent interface (MII/RMII/RGMII)
- Possibility to handle the PTP system time via an internal system time: each Ethernet controller handles it own system time.
- MDIO PHY control interface per port
- Flexible PPS generator per port

76.3 ETHSS functional description

76.3.1 ETHSS block diagram

Figure 950 shows the main components of the Ethernet subsystem.

Figure 950. Ethernet subsystem block diagram



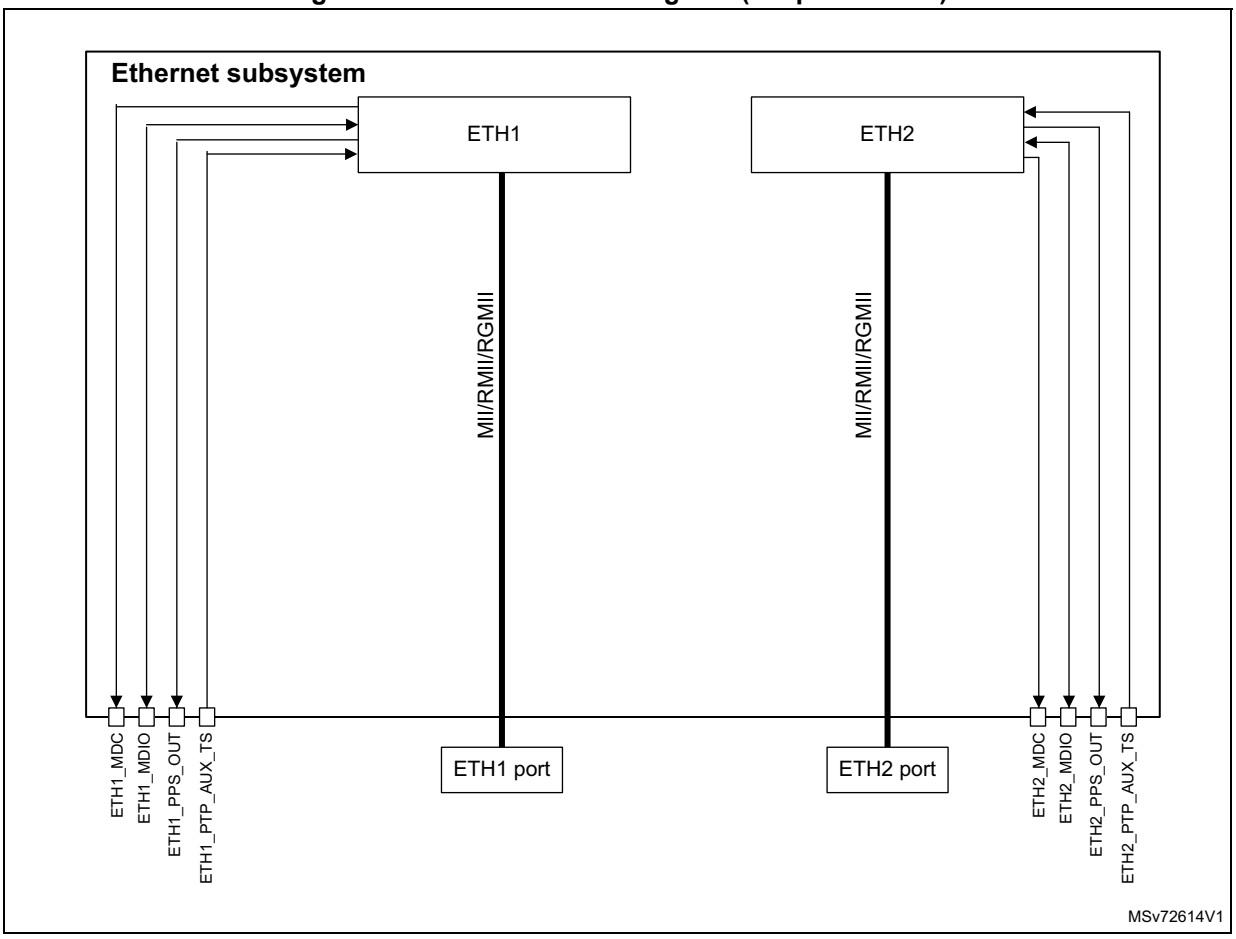
76.3.2 ETHSS pins and internal signals

The ETHSS features two Ethernet controllers (ETH1 and ETH2). They share two external ports ETH1 and ETH2.

ETH1 and ETH2 ports are directly connected to the ETH1 Ethernet controller and ETH2 Ethernet controller, respectively.

Figure 951 shows a simplified view of ETHSS pins (the MII/RMII and RGMII signals are detailed in *Section 77: Ethernet (ETH): gigabit media access control (GMAC) with DMA controller*).

Figure 951. ETHSS external signals (simplified view)



The auxiliary timestamp inputs (ETHx_PTP_AUX_TS pins) and management inputs/outputs are connected to the ETH1 and ETH2 controllers only.

PHY interface selection

The interface selection is done in the system configuration controller (SYSCFG).

The ETH1 and ETH2 port PHY interface selection is done using ETH1_SEL[2:0] in SYSCFG_ETH1CR, and ETH2_SEL[2:0] in SYSCFG_ETH2CR. This is done on the Ethernet controller side, not on the port side.

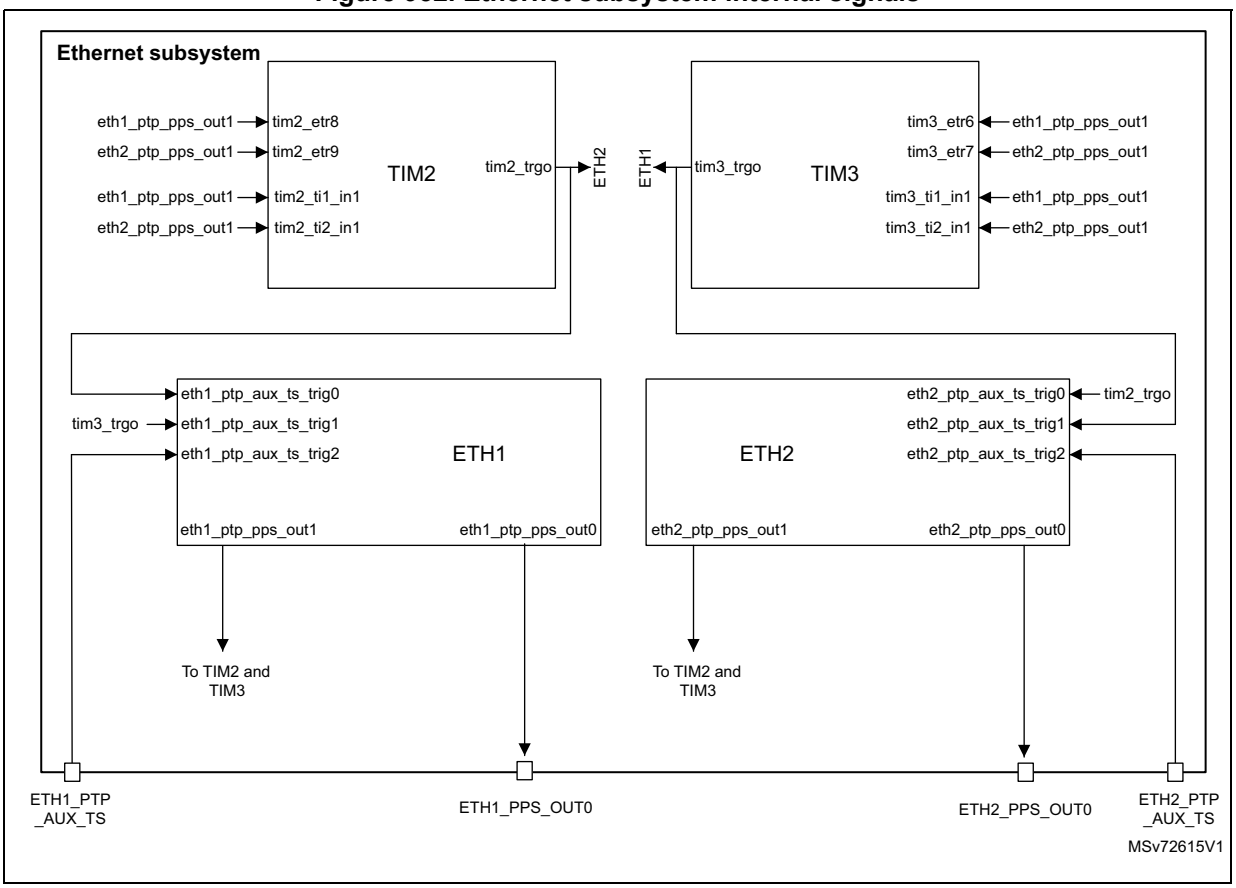
Table 795. PHY interface selection

ETH1 port	ETH2 port
ETH1 port connected to ETH1 controller and configured through ETH1_SEL2[2:0]	ETH2 port connected to ETH2 controller and configured through ETH2_SEL2[2:0]

Internal connections

The figure below represents the internal connection within the Ethernet subsystem and with the other peripherals.

Figure 952. Ethernet subsystem internal signals



Auxiliary timestamp inputs

ETH1 and ETH2 have three auxiliary timestamp inputs, which can be connected to:

- TIM2 or TIM3 tim_trgo outputs, to simultaneously timestamp in the two Ethernet controllers when each one run in a different PTP clock domain.
- ETHx_PTP_AUX_TS pins, to timestamp with an external signal (such as GPS)

PPS outputs

ethx_ptp_pps_out outputs are available off-chip as alternate functions, but can also be used internally. They are connected to the following inputs of TIM2 and TIM3 timers:

- tim_etrx pins, to serve as timer counter clock source or as a trigger to start/reset the timer (for example for PTP synchronized PWM)
- tim_tix_in pins, to measure PTP time intervals and trigger counter captures.

The PPS outputs can be configured as regular pulse-per-second outputs or in flexible PPS mode to provide a programmable train of pulses. They can also be used to synchronize software tasks on the PTP clock.

System time source selection

The IEEE 802.1AS-Rev draft defines a “system time” (also known as “gPTP clock”) that is shared among all Ethernet nodes, and one node in the system known as “grandmaster” reference. All other Ethernet nodes implement a “slave clock”, which is periodically synchronized with the “grandmaster” clock.

The Ethernet controller can be configured by software in internal system time mode. This mode assumes that the “system clock” is generated internally by the Ethernet controller. If an external “grandmaster”’ is present on the Ethernet network, it periodically synchronizes itself with it. If no “grandmaster” is present on the network, ETH1 and/or ETH2 can play this role.

The maximum value of the lower 32 bits of the system time is 999,999,999 decimal (0x3B9A C9FF) and it wraps to 0 when reaching this value (representing a full second).

76.3.3 ETHSS clocks

The clocking scheme of the Ethernet subsystem peripherals is detailed in the RCC section.

The control bits are located in the following registers:

- RCC Ethernet 1 configuration register (RCC_ETH1CFGR)
- RCC Ethernet 2 configuration register (RCC_ETH2CFGR)
- SYSCFG Ethernet1 control register (SYSCFG_ETH1CR)
- SYSCFG Ethernet1 status register (SYSCFG_ETH1SR)
- SYSCFG Ethernet2 control register (SYSCFG_ETH2CR)
- SYSCFG Ethernet2 status register (SYSCFG_ETH2SR)

77 Ethernet (ETH): gigabit media access control (GMAC) with DMA controller

77.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2015 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications, including AV nodes and TSN (time sensitive networking) nodes.

77.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

77.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2015 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- IEEE 802.1Qbv-2015, 802.1Qbu-2016, and 802.1AS-Rev D5.0 for Time-Sensitive Networking (TSN) traffic
- AMBA 2.0 for AHB slave port
- AMBA4 for AXI master port
- RGMII specification version 2.6 from HP/Marvell
- RMII specification version 1.2 from RMII consortium

Caution: The gigabit media independent interface (GMII) is only available internally to supply the RGMII adapter. No GMII signals are available off-chip.

77.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100, and 1000 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RGMII interface to communicate with an external gigabit PHY
  - RMII interface to communicate with an external Fast Ethernet PHY

- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using back pressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
  - Packet bursting and packet extension in 1000 Mbps Half-duplex operation
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII and reduced gigabit media independent interface (RGMII) PHYs
- 64-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
- Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
- Flexibility to control up to two pulse-per-second (PPS) output signals (eth_ptp_pps_outx and ETH_PPS_OUT)
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features

- Preamble and start-of-frame data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in Full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags
- Frame preemption for MAC Tx

MAC Rx features

- Automatic Pad and CRC stripping options
- Option to disable automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Four 48-bit perfect (DA) address filters with masks for each byte
  - Four 48-bit SA address comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses

- Option to pass all multicast addressed packets
- Promiscuous mode to pass all packets without any filtering for network monitoring
- Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
  - Extended VLAN tag based filtering with four filter selection
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status
- Frame preemption for MAC Rx

77.2.3 Transaction layer (MTL) features

MTL Tx and Rx Common Features

- 64-bit Transaction Layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features

- 8192-byte Transmit FIFO with programmable threshold capability
- Four Tx queues on the Transmit path with a common memory for all Tx queues
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Optional statistics related to bandwidth consumption for each queue of up to 16 blocks over a 125 µs period
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion

  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues
  - Weighted round robin (WRR)
  - Strict priority (SP)
  - Credit-based shaper (CBS)
  - Enhancement to scheduled traffic (EST)
  - Time-based scheduling (TBS)

MTL Rx features

- 4096-byte Receive FIFO with configurable threshold
- Two Rx queues on the Receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted round robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict priority (SP)

77.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 77.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 64-bit data transfers
- Separate DMA channel in the Transmit path for each queue
- Separate DMA channel in the Receive path for each queue
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors

- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Fixed-priority, weighted-strict-priority or weighted-round-robin arbitration for data read requests from multiple Transmit DMA engines
- Start and Stop modes
- Separate ports for host control access and host data interface (AXI)
- Tx DMA channels with TCP segmentation offload (TSO) feature enabled
- Routing of received packets to the DMA channels based on the DA or VLAN priority in multichannel DMA configurations
- Time-sensitive conditional packet transmission by comparing the slot time or IEEE 1588 time information provided in the descriptor (useful for AV applications)
- Programmable control for Transmit Descriptor posted writes to improve the throughput

77.2.5 Bus interface features

AXI master interface

The AXI master interface features are the following:

- Interfaces with the application through AXI4 compatible interface
- 32-bit and 40-bit address width
- 64-bit data
- OKAY, SLVERR, and DECERR responses
- Software-selected type of AXI burst (fixed and variable length burst) in AXI master interface; extended length fixed bursts of 32, 64, 128, and 256
- Handshaking on AXI read and write data channels

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry

77.2.6 Audio and video features

The Ethernet peripheral can be used in Audio Video (AV) mode. The supported features are compliant with the industry standards for AV traffic:

- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- Multiple Rx queues on the Receive paths for AV traffic, and multiple Tx queues on the Transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues
- Programmable Slot Interval with range from 1 µs to 4096 µs and granularity of 1 µs.

77.2.7 Time sensitive networking features

The Ethernet peripheral supports the following time sensitive networking (TSN) features:

- IEEE 802.1Qbv-2015, Enhancements to Scheduling Traffic
- IEEE802.1Qbu/802.3br, Frame preemption and Interspersing Express Traffic

77.2.8 Generic queuing features

The Ethernet peripheral supports the following features for generic queuing:

- Programmable control for routing Receive packets with Multicast/Broadcast destination address to a programmable Receive queue
- Support routing of untagged Receive packets to a programmable Receive queue
- Programmable control for routing VLAN tagged and untagged IEEE 1588 PTP over Ethernet Receive packets to a same or separate programmable Receive queue
- Programmable control for routing Unicast/Multicast Receive packets that fail the destination address filter to a separate programmable Receive queues

77.3 Ethernet pins and internal signals

Table 796 lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII, RMII or RGMII) and on the device configuration.

Table 797 shows the internal Ethernet signals.

Table 796. Ethernet peripheral pins

Port name	Digital port type	Description
ETH_COL	Input	Collision detection signal, MII only.
ETH_CRS	Input	Carrier sense signal, MII only
ETH_REF_CLK	Input	RMII reference clock
ETH_RX_CLK	Input	MII/RGMII timing reference for Rx data transfers
ETH_RXD[3:0]	Input	Receive data. 4 pins for MII and RGMII, 2 for RMII.
ETH_RX_DV	Input	Receive data valid

Table 796. Ethernet peripheral pins (continued)

Port name	Digital port type	Description
ETH_CRS_DV	Input	RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
ETH_RX_CTL	Input	After demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).
ETH_RX_ER	Input	Receive error
ETH_TX_CLK	Input	MII timing reference for Tx data transfers
ETH_GTX_CLK	Output	RGMII timing reference for Tx data transfers
ETH_TXD[3:0]	Output	Transmit data. 4 pins for MII and RGMII, 2 for RMII.
ETH_TX_EN	Output	Transmit data enable
ETH_TX_CTL	Output	Multiplexing of ETH_TX_EN on the rising edge of ETH_TX_CLK, and a logical XOR between ETH_TX_EN and ETH_TX_ER values on falling edge of ETH_TX_CLK
ETH_TX_ER	Output	Transmit error
ETH_MDC	Output	Management data clock
ETH_MDIO	Input/output	Management data
ETH_PHY_INTN	Input	PHY interrupt
ETH_PPS_OUT	Output	PTP pulse-per-second output 0
ETH_PTP_AUX_TS(1)	Input	Trigger input for the auxiliary snapshots of the PTP system time

1. This pin is not available on all devices.

Table 797. Ethernet internal input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_aclk	Digital input	AXI clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wake-up packet is received
eth_tx_clk	Digital input	Tx kernel clock
eth_rx_clk	Digital input	Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock
eth_ptp_pps_out1	Digital output	PTP pulse-per-second signal
mac_speed_o[1:0]	Digital output	MAC speed information used by the RCC
clk_ptp_ref_i	Digital input	PTP reference clock input

Table 797. Ethernet internal input/output signals (continued)

Signal name	Signal type	Description
eth_ptp_trig[3:0]	Digital input	Trigger input for auxiliary snapshots of the PTP system time
ptp_timestamp_i[63:0]	Digital input	External PTP timestamp bus

77.4 Ethernet architecture

The Ethernet peripheral is composed of 4 main functional modules:

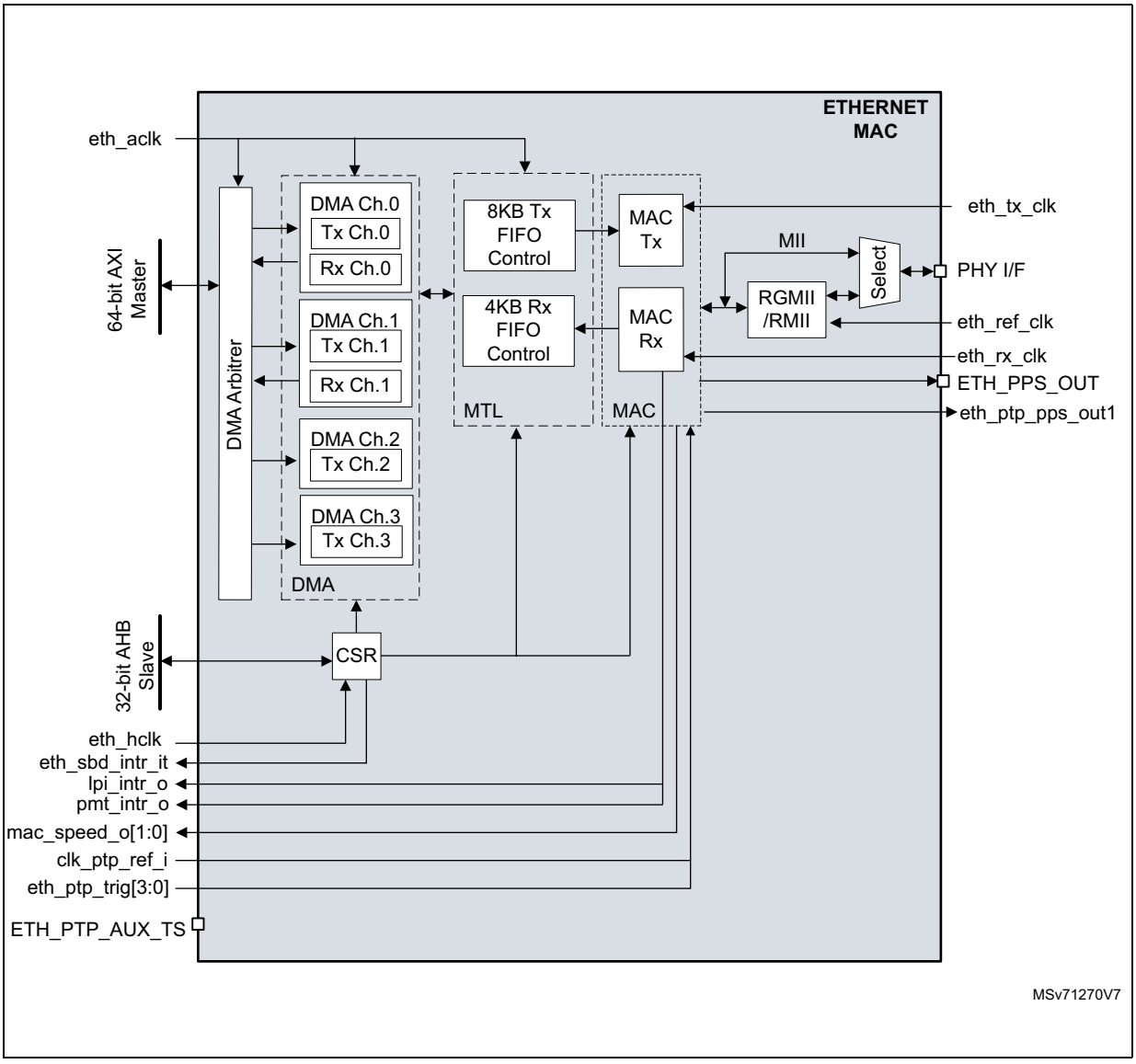
- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with two physical channel for reception and four for transmission. It controls the data transfers between MAC and system memory through the AMBA AXI4 64-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII or the RGMII PHY Media Independent Interfaces.

Caution: The gigabit media independent interface (GMII) is only available internally to supply the RGMII adapter. No GMII signals are available off-chip.

Figure 953. Ethernet high-level block diagram



1. For a definition of the internal signals, refer to *Table 797*.
2. Refer to RCC chapter "Clock distribution for Ethernet" for a detailed description of the Ethernet clock architecture.
3. The ETH_PTP_AUX_TS pin is not available on all devices.

77.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*) and *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*.

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*.

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

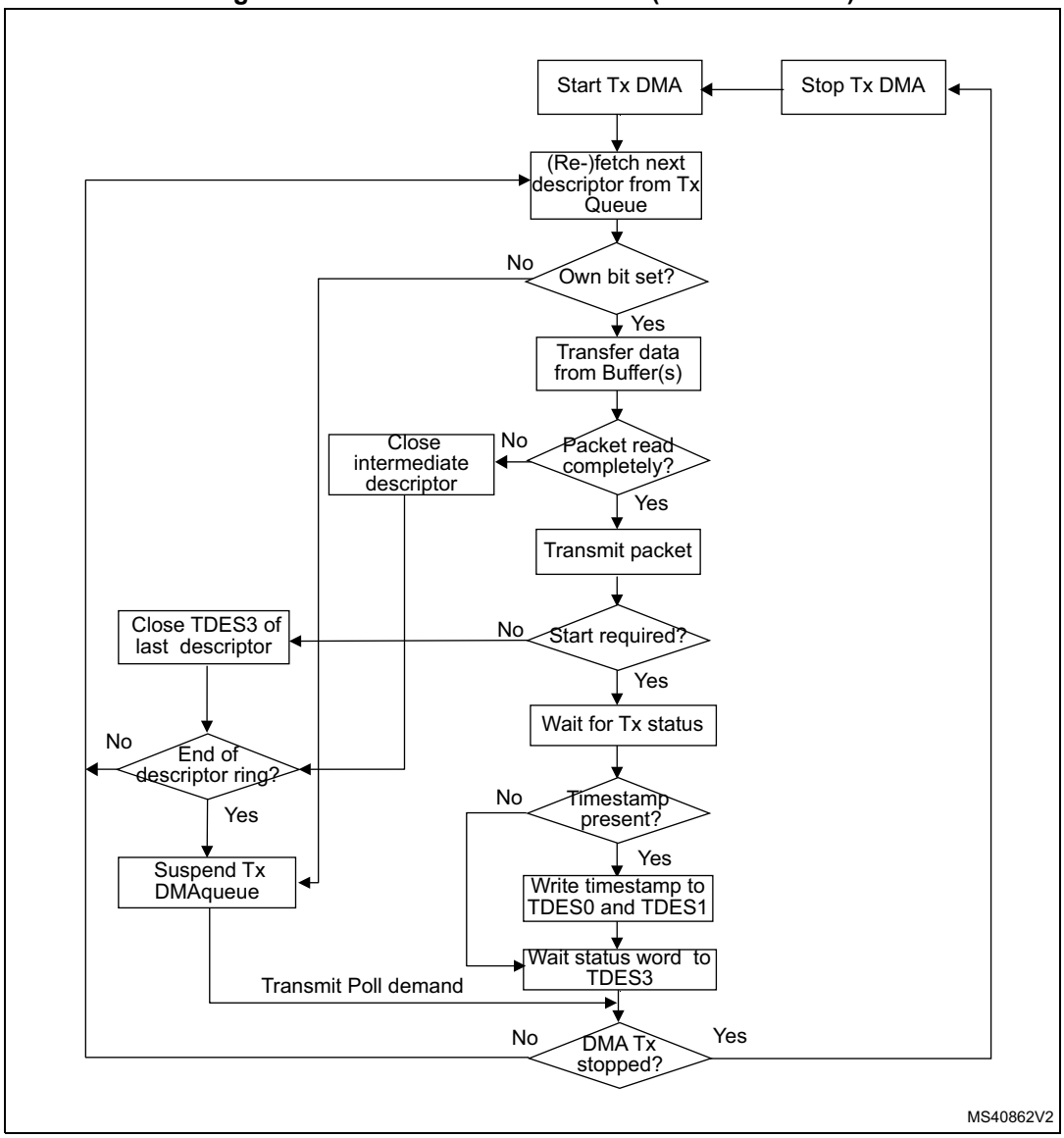
Descriptors are specified in *Section 77.10: Descriptors*.

DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the Transmit channel.
3. While in the Run state, the DMA runs an arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   – The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   – The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   – An error condition occurs.
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
10. Bit 0 of *Channel x status register (ETH_DMACxSR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 954. DMA transmission flow (standard mode)



DMA transmission in OSP (Operate on Second Packet) mode

In Run state, if bit 4 is set in the *Channel x transmit control register (ETH_DMACxTXCR)*), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

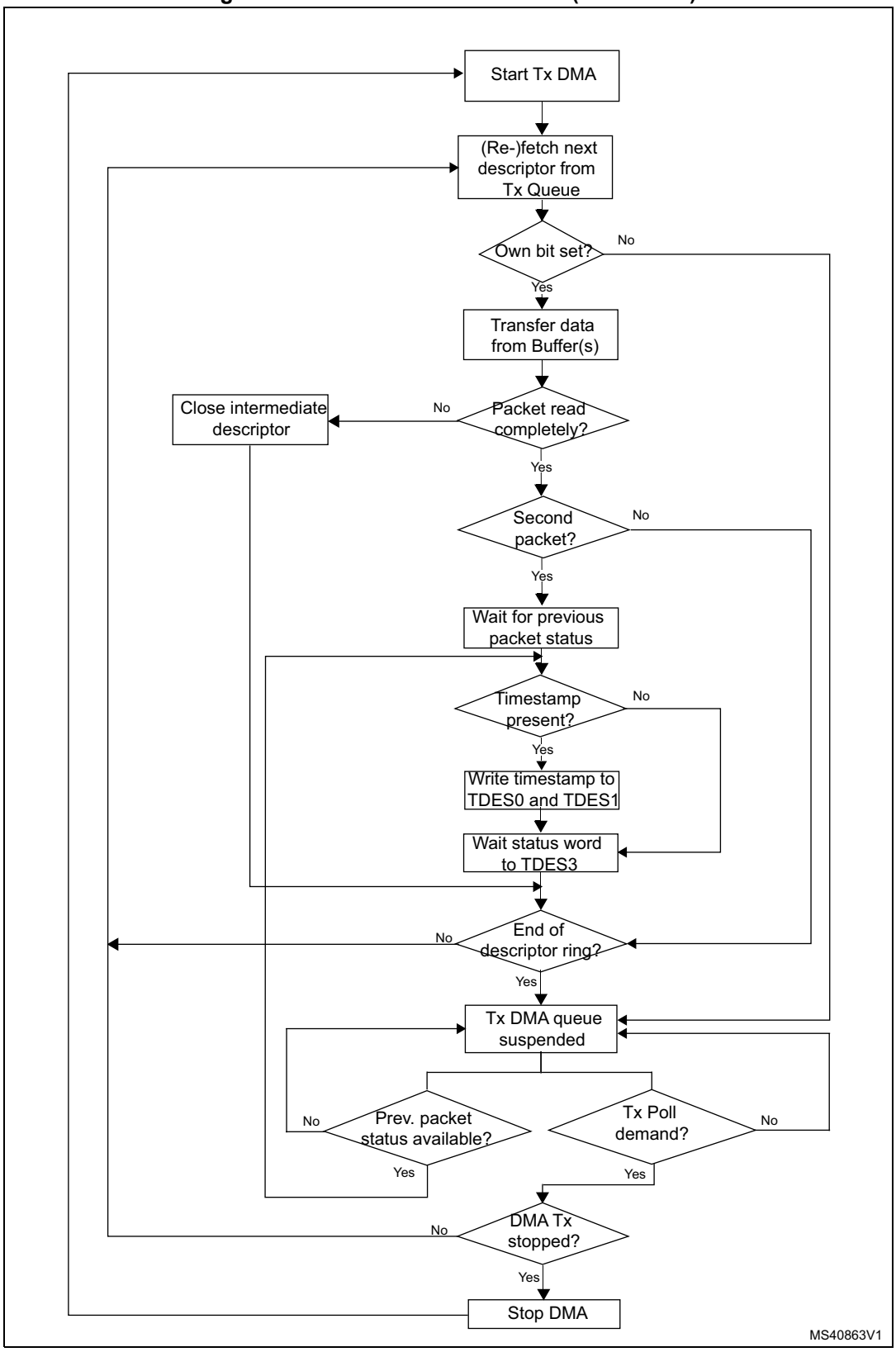
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 956: Receive DMA flow*.

Figure 955. DMA transmission flow (OSP mode)



DMA reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

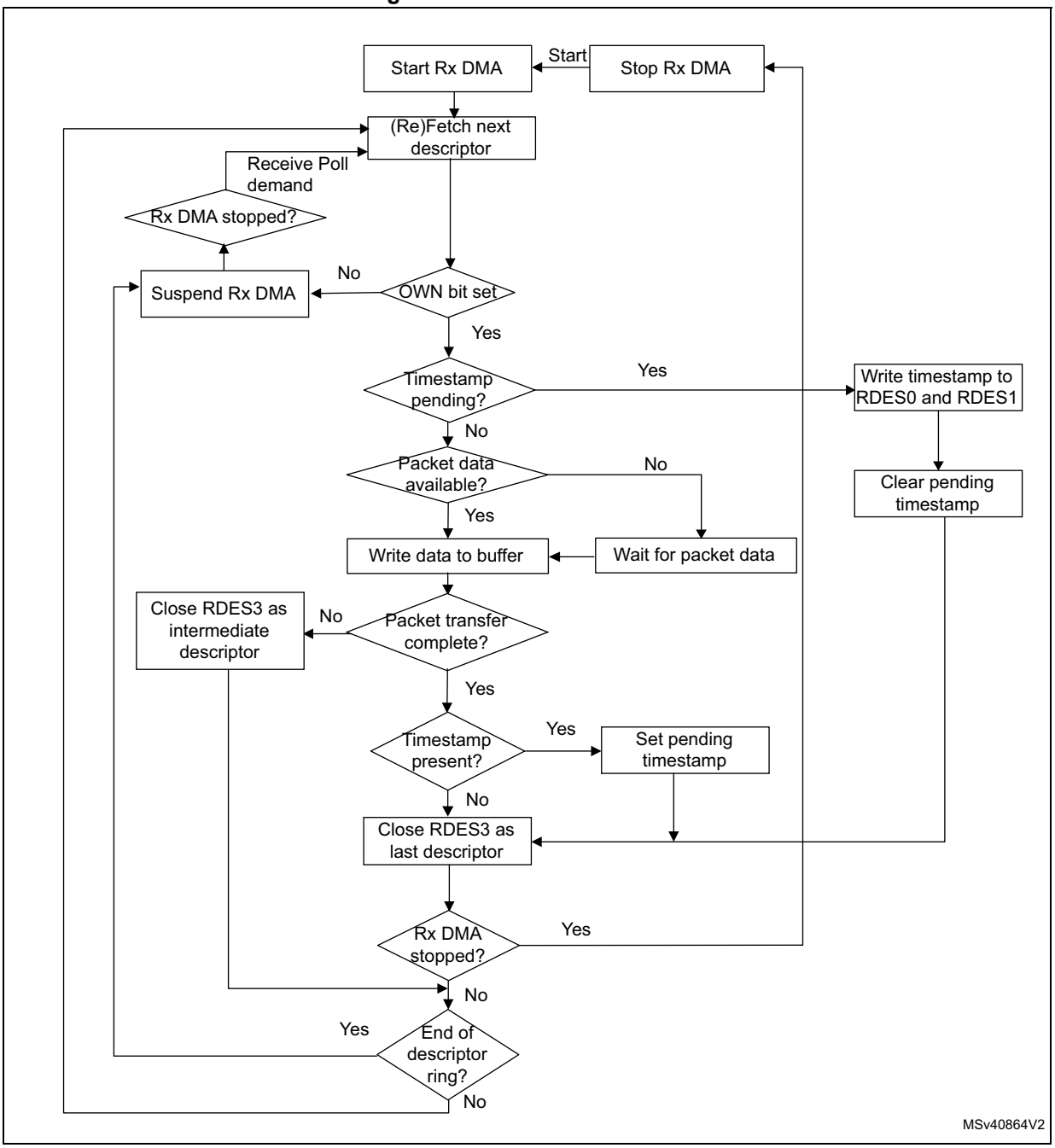
The DMA Rx descriptor ring structure is described in *Section 77.10: Descriptors*.

The reception sequence for Rx DMA engine is as follows (see also *Figure 956: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.
2. When bit 0 of *Channel x receive control register (ETH_DMACxRXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given, and the application updates the channel Receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

Note: Refer to Section : Descriptor tail pointer handling examples for updating the correct value in receive descriptor tail pointer register.

Figure 956. Receive DMA flow



Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the *DMA mode register (ETH_DMAMR)*.

77.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 64-bit wide data paths: the Transmit path and the Receive Path.

- Transmit path

  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue in Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 8192 bytes.

- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in the operating mode register of the corresponding queue), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 4096 bytes.

77.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the GMII or MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     the packet was successfully transmitted;

     the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.

– If any one of the following event happens, the MAC aborts the packet transmission:

no carrier (Half-duplex mode)

loss of carrier (Half-duplex mode)

excessive deferral (Half-duplex mode)

late collisions (Half-duplex mode)

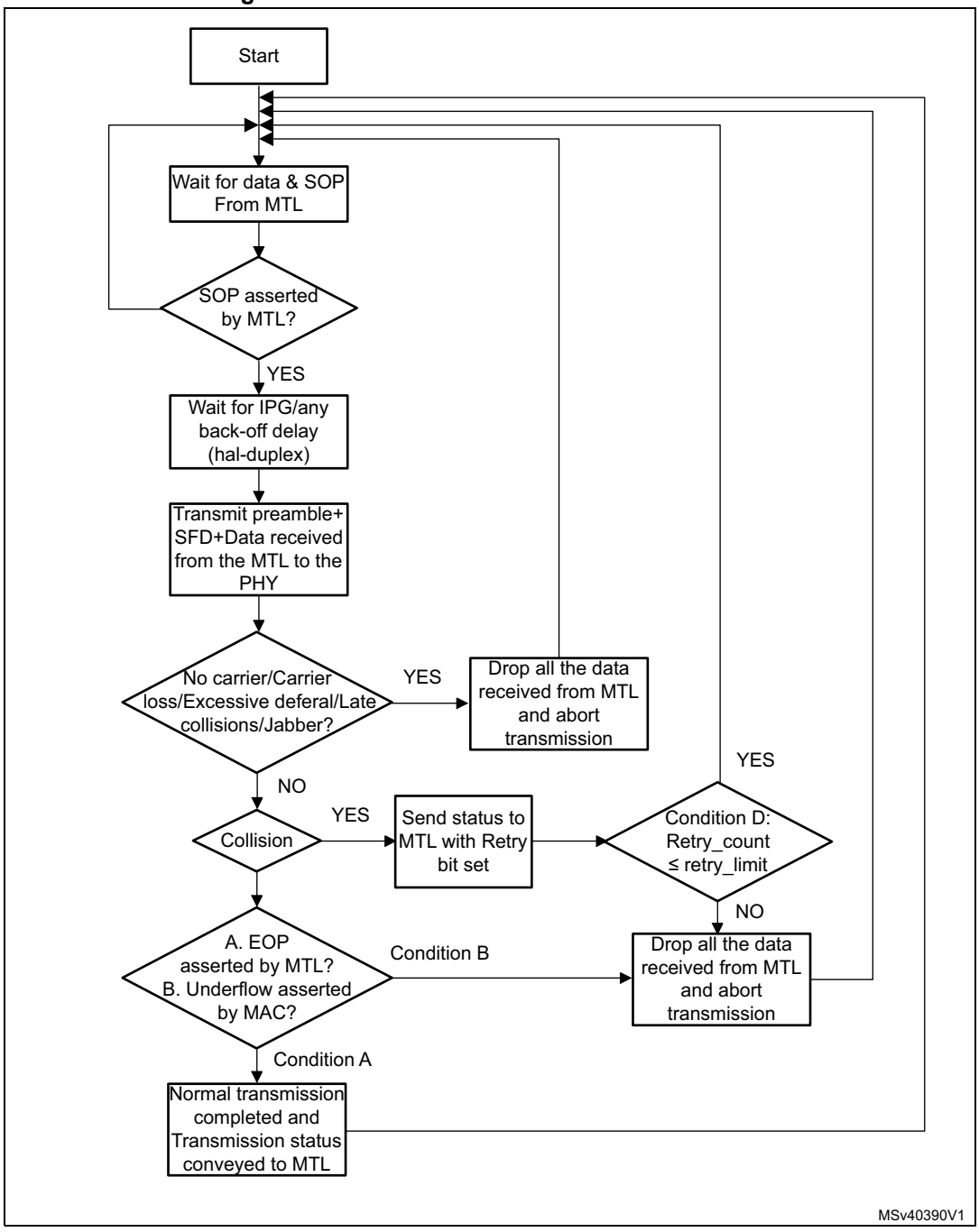
jabber

the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 957: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 957. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on GMII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

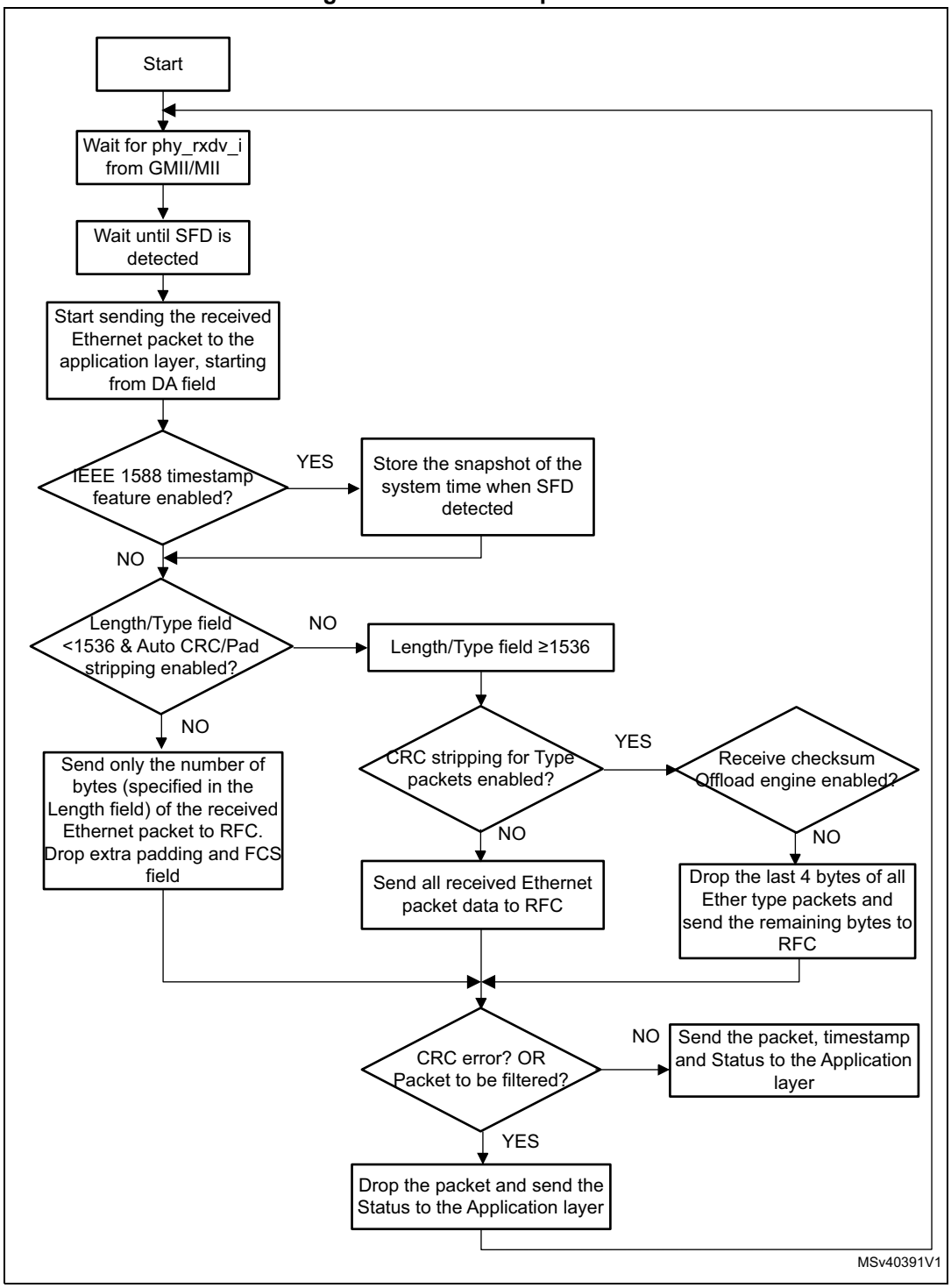
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of GMII or MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0x5D byte).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on GMII or MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application.The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
4. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
5. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog and jabber timeout register (ETH_MACWJBTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 958. MAC reception flow



77.4.4 Multiple channels and queues support

This chapter describes how the Ethernet peripheral core supports multiple queues and channels.

Two queues on Rx side and four queues on Tx side are supported. Four DMA channels are implemented on Tx side, one channel per queue. Two DMA channels are implemented on Rx side, one channel per queue.

The multiple queues enable to support the Audio video bridging (AVB) protocol.

Multiple queues and channels support in the Transmit path

The number of Tx DMA channels is equal to the number of Tx queues, and is direct one-to-one mapping. This enables the interleaving of packet/data transfer of multiple Tx DMA on the host/system bus, and each Tx queue is reserved for the corresponding Tx DMA. Therefore, the integrity of packet transfer by each Tx DMA is maintained in the Tx queue and enables efficient utilization of the host bus for data transfers among multiple Tx DMA without depending on the packet length, Transmit buffer availability and so on.

If two Tx DMAs are allowed to transfer data into the same Tx queue, when a Tx DMA starts a packet transfer, the other Tx DMA cannot transfer data to the same Tx queue unless the previous packet transfer is complete. This means that the second DMA remains idle until the first packet transferred is complete, which is effectively a one-to-one mapping of the Tx DMA to the Tx queue.

The Ethernet peripheral supports multiple Tx channels. The priority scheme between both channels is fixed priority. In fixed priority scheme, the channel with the highest priority always wins the arbitration when it requests the bus, as shown in *Table 798*.

Table 798. Fixed priority scheme for DMA channels

Priority level	Channel
0	Channel 0
1	Channel 1
2	Channel 2
3	Channel 3

The fixed priority scheme does not necessarily cause the “starving” of lower priority channels for the AXI bus because of the capabilities of the AXI protocol and characteristics of the TxDMA, described as follows:

- Read command requests from a TxDMA for a burst of data transfers are driven on a separate AXI channel, and their execution takes one clock cycle. The actual data

transfer starts on a separate AXI channel only after the request is accepted. It takes several clock cycles to complete with relatively large read access latencies.

- The AXI master is free to issue subsequent requests even before the previous data transfers are complete.
- A TxDMA requests another data transfer only after the data transfer corresponding to its previous request is complete and transferred to the MTL Tx queue and space is available in Tx queue to accept the next request.
- The Ethernet peripheral can be programmed to control the maximum number of outstanding requests, requested burst sizes by DMA, and the burst sizes of each data transfer on the AXI bus.

Considering the previous points, the Ethernet peripheral can be programmed so that the requests of all TxDMAs are placed and executed on the AXI bus before the data transfer of the first TxDMA is complete and it raises the next request to the arbiter. For example, in a 64-bit AXI data bus, proceed as follows:

1. TxDMA PBL = 64 (512 bytes).
2. Number of outstanding AXI read requests = 16.
3. Maximum burst length on AXI (BLEN) = 32 (256 bytes).
4. Number of active TxDMAs = 8.

Assuming that all TxDMAs place a request simultaneously and get serviced by the fixed priority algorithm, all the requests are placed on the AXI bus in a window of 32 clocks (two clocks per AXI request, two AXI burst requests per TxDMA request). The data transfer of the first request takes 32 clock cycles + the read-access latencies (assuming that there are tens of cycles). So, the TxDMA arbiter is already IDLE and has completed servicing all the TxDMA requests before this data transfer completes.

Effectively, all the TxDMA channels get equal access to AXI bus and the corresponding Tx queues get filled up at the same rate. The data gets read out from the Tx queues as per the traffic class scheduler in the MTL. So at a steady state, the bandwidth for each channel is controlled by the priorities and settings of the traffic scheduler in the MTL.

Programming guidelines for multiple queues and channels in the Transmit path

See *Section 77.9.8: Programming guidelines for multichannel multiqueue operation on page 4445*.

Multiple queues and channels support in the Receive path

The Ethernet controller supports independent selection of the number of Rx DMA channels and the number of Rx queues in the MTL Rx buffer. Therefore, the packets from any Rx queue can be serviced by any of the Rx DMA. As all the packets of Rx queues are stored in a shared RxFIFO buffer in MTL, the Rx scheduler selects the Rx queue whose packet data are to be transferred to the destination Rx DMA, which then forwards the packet data to the host memory. The sequence of events in time is as follows:

1. Each Rx DMA indicates whether it is ready to transfer data, that is, when the Rx DMA has fetched the descriptor and has a receiver buffer ready to accept the packet data. Each Rx DMA also indicates the number of words that it can transfer, based on the

space available in the Rx buffer and the value programmed in the RXPBL[5:0] bitfield of its *Channel x receive control register (ETH_DMACxRXCR)*.

2. The MTL Rx scheduler matches the number of bytes/packets available for transfer at the top of all Rx queues with the corresponding Rx DMA ready status and the number of bytes requested by it.
3. If there are multiple successful matches, the scheduler selects the Rx queue to be serviced, based on the arbitration schemes (Strict priority or Weighted Strict priority) decided by the RAA field of the *Operating mode register (ETH_MTLOMR)*. In the Weighted Strict priority scheme, the weight of each Rx queue is set in the RXQ_WEGT[2:0] bitfield of the corresponding *Rx queue x control register (ETH_MTLRXQxCR)*.
4. The MTL Rx controller fetches the data of the selected Rx queue from the shared RxFIFO memory and starts the transfer of the requested PBL number of bytes to the DMA block or until the end of packet and the corresponding Rx status is transferred (whichever is first).
5. After completing the current request, the scheduler repeats the process by starting a new arbitration cycle. However, it selects the same Rx queue for servicing, if the end of packet is not transferred and the RXQ_FRM_ARBIT field of the *Rx queue x control register (ETH_MTLRXQxCR)* is 1, for the Rx queue.

Rx queue to DMA mapping

The packets in the MTL Rx queues can be routed to any of the multiple DMA channels, by programming the Rx queue and the *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)*.

Two types of Rx queue-to-DMA mapping are possible through programming: static mapping and dynamic (per packet) mapping.

- Static mapping

  In Static mapping mode, all the packets of an Rx queue are connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH and Q0DDMACH of the *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)*.

  Similarly, packets from other Rx queues can be routed to any DMA channel by programming the register fields corresponding to each queue.

- Dynamic (per packet) mapping

  In Dynamic (per packet) mapping mode, the destination DMA channel is decided by the MAC core receiver for each packet.

  In this mode, the destination DMA channel of a packet being read from a Rx queue is not constant but decided independently for each packet. For example, by setting the Q1DDMACH bit of the *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)*, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC receiver for each packet, depending on the following, in decreasing order of priority:

  – L3-L4 filter based DMA selection

    When the Layer 3 and Layer 4 packet filtering is enabled, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programmed and enabled for comparison in the MAC L3 and L4 address control register. If the match is successful, the DMA channel number programmed in the DMCHNx bits of the *L3 and L4 control 0 register (ETH_MACL3L4C0R)* and

L3 and L4 control 1 register (ETH_MACL3L4C1R) is selected as the destination DMA channel number provided DMCHEN bit of the same register is set.

If none of the L3-L4 registers give a comparison match, the Ethernet peripheral proceeds to the next step.

– Extended VLAN based DMA selection

Extended routing is applicable only if the VLAN filter has passed. Routing is only based on perfect filter result. Each perfect filter has a DMA channel enable and a DMA channel number field which can be programmed. Routing is applicable for a filter only if the DMA channel enable bit is set.

The frame is routed to the smallest matching filter DMA channel provided it is enabled. If that filter DMA channel number is not enabled, the frame gets routed to Channel 0. For example, if a frame VLAN tag matches filter 1, then the MAC checks if filter 1 DMA channel number is enabled through programming. If yes, the frame gets routed to the programmed value; otherwise it gets routed to DMA. When the inverse filter is enabled, the frame is routed to the least mismatched filter DMA channel number provided it is enabled. If the DMA channel enable bit is not set, then the frame is routed based on DA-based addressing or to Channel 0.

If Hash filter is also enabled, it is used to determine the filter result only. Routing still depends on the enabled perfect filters. If none of the perfect filters are enabled or if all of them are bypassed, the VLAN filter based routing is not done. The frame is routed through DA-based addressing or to Channel 0. If all the perfect filters give a fail result and the Hash filter has passed, VLAN filter result is a pass. However, routing is based on DA-based addressing or to Channel 0. The behavior is similar when the inverse filtering is enabled.

– Ethernet DA-based DMA selection

The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number.

If none of the previous operations is able to make a successful match/decision, then the packet is routed to DMA Channel 0 by default.

Programming guidelines for multiple queues and channels in the Receive path

See *Section 77.9.8: Programming guidelines for multichannel multiqueue operation on page 4445*.

Selection of the tag priorities assigned to Rx queues

The VLAN tag priorities can be assigned to Rx queues by programming the PSRQ field in the corresponding *Rx queue control 2 register (ETH_MACRXQC2R)*. The bit corresponding to the VLAN tag priority can be set in the PSRQ field to assign the corresponding priority to the Rx queue. As an example, to assign VLAN tag priority 3 to Rx queue 0, set bit3 of PSRQ field in the *Rx queue control 2 register (ETH_MACRXQC2R)*. The VLAN tag priority assigned to a given Rx queue must be unique. This means that multiple Rx queues cannot be assigned the same VLAN tag priority. However, multiple VLAN tag priorities can be assigned to the same Rx queue.

The settings in the PSRQ field is used for VLAN tagged Rx packet routing to Rx queues as well as for PFC based Tx flow control. The received VLAN tagged Rx packet is routed to the Rx queue that matches the VLAN tag priority. In PFC-based Tx flow control, the PSRQ field corresponding to a particular Rx queue is used to enable VLAN tag priorities in the PFC packet that is transmitted when the corresponding Rx queue threshold levels are reached.

Distribution of Rx packets from MAC to Rx queues

The Ethernet peripheral provides multiple Rx queues to support classification and distribution of ingress packets. The MAC receiver pushes the ingress packets to the Rx queues based on packet type (such as tagged, untagged, multicast, PTP, AVB), tag priorities, MAC packet filter results, preemptable or express packet, and so on. The distribution is controlled by the settings of:

- *Rx Queue control register (ETH_MACRXQCR)*
- *Rx queue control 0 register (ETH_MACRXQC0R)*
- *Rx queue control 1 register (ETH_MACRXQC1R)*
- *Rx queue control 2 register (ETH_MACRXQC2R)*
- *Packet filtering control register (ETH_MACPFR)*

Additionally, the Ethernet peripheral supports distribution of ingress packets based on user-defined Type field of the ingress packets when TBRQE bit of the *Rx queue control 1 register (ETH_MACRXQC1R)* is enabled. Up to eight types for matching can be programmed in the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* which is accessible through *MAC Indirect Access Control register (ETH_MAC_IACR)*. The ETH_MAC_TMRQR register contains

- TYP[15:0] field: 16-bit type
- TMRQ[2:0] bitfield: Rx queue number to which ingress packet is pushed when its Type field matches TYP.
- PFEX field: this bit indicates whether the TYP is applicable for express packets or preemptable packets.

Based on the packet filter settings, the MAC receiver decides either to drop or forward the ingress packet (for more details, see *Section 77.5.4: Packet filtering*). However, the results of Layer 3 and Layer 4 filtering, mentioned in *Section : Layer 3 and Layer 4 filtering*, are not considered for the distribution of ingress packets into the Rx queues. Only the results of DA/SA/VLAN filters mentioned in *Section : MAC source or destination address filtering* and *Section : VLAN filtering* are applicable.

The filtering of the broadcast packets is exclusively controlled by the DBF bit of the *Packet filtering control register (ETH_MACPFR)*. The filtering of multicast packets is controlled by the PM bit of the *Packet filtering control register (ETH_MACPFR)*, in addition to the DA/SA match.

By default, packets that fail any of the DA/SA filters, are dropped by the MAC receiver. However, packets that fail the DA/SA/VLAN filter can still be forwarded to the application when the RA field of the *Packet filtering control register (ETH_MACPFR)* is 1. Even when RA field is 0, VLAN filter fail packets can be forwarded if the VTFE field of the *Packet filtering control register (ETH_MACPFR)* is 0.

The ingress packets go through the following operations in decreasing order of priority.

1. RA = 0: ingress packets are dropped if any of the following conditions is true.
   - DA/SA filter fail
   - VLAN filter fail and VTFE field of the *Packet filtering control register (ETH_MACPFR)* is 1
2. RA = 1: ingress packets that fail the DA/SA filter are also forwarded to the Rx queue as mentioned in *Table 799*.

Table 799. Routing DA/SA failed packets to Rx queue

Packet type	TSN type	Destination Rx queue determined by
Broadcast/Multicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
Unicast	Express	if UFFQE = 1, UFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
All	Preemptable	FPRQ field

3. RA = 1 and VTFE = 0: ingress tagged packets that pass the DA/SA filter but fail the VLAN filter are forwarded to the Rx queue as mentioned in *Table 800*.

Table 800. Routing DA/SA pass but VLAN filter fail packets to Rx queue

TSN packet type	Destination Rx queue determined by
Express	if VFFQE = 1, VFFQ; else RxQ0
Preemptable	RQ field

4. Ingress packets that pass the DA/SA/VLAN filters are forwarded to the Rx queue.
   – *Table 801* shows the routing priority in the decreasing order when OMCBCQ field of the *Rx queue control 0 register (ETH_MACRXQC0R)* is 0.

Table 801. Priority routing of filter pass packets when OMCBCQ = 0

Packet type	TSN type	Tagged	Destination Rx queue determined by
Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast AVTP control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
Unicast PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on the value of TPQC.
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

   – *Table 802* shows the routing priority in the decreasing order when OMCBCQ field of the *Rx queue control 1 register (ETH_MACRXQC1R)* is 1.

Table 802. Priority routing of filter pass packets when OMCBCQ = 1

Packet type	TSN type	Tagged	Destination Rx queue determined by
All AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
All PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on TPQC
Remaining Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Remaining Unicast	Express	No	if TBRQE = 1, TBRQ; else UPQ
		Yes	if TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	if TBRQE = 1, TBRQ; else FPRQ
		Yes	if TBRQE = 1, TBRQ; else PSRQ

Note: The fields mentioned in Table 799 through Table 802 belong to Rx queue control 0 register (ETH_MACRXQC0R), Rx queue control 1 register (ETH_MACRXQC1R) and Rx queue control 2 register (ETH_MACRXQC2R).

Note: If the RxQ pointed by the various fields in Table 799 through Table 802 is not enabled in the Rx queue control 0 register (ETH_MACRXQC0R) as per the packet type, the packet is written to

- *RxQ0 for Express packets*
- *FPRQ field for Preemptable packets*

If RxQ0/FPRQ is not enabled, the packet is dropped.

Note: If the AV8021ASMEN field of the Timestamp control Register (ETH_MACTSCR) is 1, MAC receiver checks only for untagged PTP packet. Packets tagged with PTP payload are considered as Remaining (Unicast/Broadcast/Multicast) tagged packets in Table 799 through Table 802.

The priority field of the outer VLAN tag is considered for routing.

AV feature

The Ethernet peripheral supports the AV data transfer in 100-Mbps and 1000-Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: enables the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration

and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.

- IEEE 802.1Qat-2009: enables the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 Full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

While the AVB standards were originally designed for audio and video over standard Ethernet, extending these to control streams would additionally benefit the industrial and automotive sectors. Time sensitive networking (TSN) is a set of standards developed by the Time Sensitive Networking Task Group with the intent to extend the future AVB enhancements to other sectors that could benefit.

Bridges are increasingly used to interconnect devices that support scheduled applications (for example, industrial automation, process control and vehicle control). The IEEE 802.1Qbv-2015 (Enhancements to Scheduling Traffic) provides performance assurances of latency and delivery variation to enable these applications in an engineered LAN while maintaining the existing guarantees for the credit-based shaper and best-effort traffic.

The IEEE 802.3br (Interspersing Traffic) and IEEE 802.1Qbu (frame preemption) enables the suspension of a large preemptable packet being transmitted by the MAC layer to enable one or more express packets to be transmitted before the transmission of the preemptable packet is resumed. This provides the capability to schedule express traffic packets with minimal delays/latencies or at predictable times with efficient utilization of the line bandwidth.

As per the IEEE specification for 802.1Qbv, 802.1Qbu, and 802.3br, the TSN features are supported only in the following modes:

- Full duplex
- 100 Mbps or higher link speed
- Pause or no flow control
- When Qbv schedule is enabled for time-sensitive traffic, Pause or PFC is likely to interfere with the Qbv schedules. So, it is recommended not to enable PFC/PAUSE when Qbv scheduler is enabled.
- Multiple Tx queues (for Qbv) enabled
- One or more Express queues and one or more preemption queues on both transmit and receive (for Qbu/802.3br) enabled.

Transmit path functions

When the AV feature is selected, the Transmit paths of Tx queues are enabled by default.

The Transmit path of queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The Transmit path of the AVB Tx queue supports traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets.
- The credit for the queue is positive as per the algorithm.

When the credit-based shaper algorithm is disabled for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for queue 1, the queue is selected if the packet is available in the channel and has a positive or zero credit.

If the credit-based shaper algorithm is disabled for a given Tx queue, the packet to be transmitted is selected based on the fixed priority scheme as described in *Table 803*.

Table 803. Weight for DMA channels

TCW field	Transmit channel weight
000	1
001	2
010	3
011	4

Receive path functions

When the AV feature is enabled, the Receive path of Queue 0 is enabled by default. The whole traffic is received on this channel. The Receive paths of additional queues can be enabled. By enabling the Receive paths of multiple channels, the received data can be demultiplexed to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if the received packet is an AV packet as well as the corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the Receive descriptor as explained in *Section 77.10.4: Receive descriptor*. All received packets with an EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets

  AV data packets are always tagged. Tagged AV data packets are received following the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits[15:8] in the transmit flow control register of the corresponding queue.

- AV control packets

  The AV control packets can be either tagged or untagged. By default, untagged AV control packets are received on queue 0. To receive these packets on queue 1, program bits[2:0] of the *Rx queue control 1 register (ETH_MACRXQC1R)*. Similarly to the AV data packets, tagged AV control packets are received following the programmed priority value.

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on queue 0. To receive these packets on any other queue, program bits[6:4] of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

Credit-based shaper algorithm

The Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in queue 0. Queue 1 can be programmed to use the credit-based shaper algorithm.

The following sections provide information on the credit-based shaper algorithm implementation:

- Credit value
- idleSlopeCredit and sendSlopeCredit values
- Bandwidth status

Credit value

The credit value is accumulated every transmit clock cycle, that is every 40 ns in 100-Mbps mode and every 8 ns in 1000-Mbps mode. The credit to be added or subtracted per cycle can be fractional depending on the required idleSlope and sendSlope values, as described in *Table 804*.

Table 804. Example of credit value per transmit cycle

Mode	portTransmitRate, idleSlope, sendSlope	Credit value
100 Mbps	– portTransmitRate = 100 Mbps – idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) – sendSlope = 30 Mbps	– credit = 2.8 bits, accumulated per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. – credit = 1.2 bits, drained per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO depending on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA application bus.

The credit for a queue builds up only when the packet is available, but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet peripheral supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. This mode can be enabled through the CC bit of the CBS control register (*Tx queue x ETS control register (ETH_MTLTXQxECR)*.

- When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting.
- When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024 (such as 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229). In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register (*Tx queue x ETS status register (ETH_MTLTXQxESR)*) gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in bits[6:4] of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, queue 1 and queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and queue 0 uses the strict-priority algorithm. If queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

Slot number function

When the AV feature is enabled, the slot number function can be used to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

The slot number at which the DMA should fetch the data from system memory can be programmed in the Transmit Descriptor Word 3 (TDES3). This 4-bit field enables the application to schedule data fetching at up to 16 slots with a programmable slot interval that ranges from 1 µs to 4096 µs and a granularity of 1 µs. This field is applicable only to the AV channels.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated based on a programmable slot interval (with range from 1 µs to 4096 µs) of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, when the subsecond counter rolls

over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until a match is found.

To enable the DMA to fetch data only if it matches the current slot or the next two slots, program bit 1 of the *Channel x slot function control status register (ETH_DMACxSFCSR)* of the corresponding DMA channel.

If the slot number in the descriptor is less than the reference slot number, the DMA considers it as a future slot.

The slot number check can be enabled by setting bit 0 in the *Channel x slot function control status register (ETH_DMACxSFCSR)* of corresponding DMA channel. If this check is disabled, the packets are fetched immediately after the descriptor is read. In addition, bits [19:16] indicate the value of the reference slot number in DMA.

Programming guidelines for AV feature

See *Section : Initializing the DMA on page 4453*.

See *Section : Enabling slot number checking on page 4454*.

See *Section : Enabling average bits per slot reporting on page 4454*.

Queue modes

A Tx queue can be enabled for generic or AV traffic. When multiple Tx queues are enabled for generic traffic, queuing is based on WRR or WSP algorithms.

When a Tx queue is enabled for AVB traffic, queuing is based on CBS or SP algorithms.

A Rx queue can be enabled for generic or AV traffic. A particular queue enabled for generic or AV based routing is determined by the RXQ0EN and RXQ1EN fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.

Below an explanation on how Rx queues are enabled based on the selected features:

- Multiple Rx queues

When multiple Rx queues are selected, all queues are enabled for generic queuing based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. By default untagged packets are routed to Receive queue specified in UPQ field of *Rx queue control 1 register (ETH_MACRXQC1R)*. Queue 0 is the default value of UPQ[2:0] bitfield. The default value can be overridden with any other value, for the UPQ[2:0] bitfield. The Rx packets can also be routed to a particular DMA channel based on the DCS field of the perfectly-matched MAC Address register (*MAC Address x high register (ETH_MACAxHR)*.

- Multiple Rx queues with AV feature

When AV is enabled for all selected Rx queues, the queuing is done based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register (*MAC Address x high register (ETH_MACAxHR)*). The AV control packets (tagged or untagged) are routed based on the AVCPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*. The PTP over Ethernet packets are routed based on the PTPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

77.5 Ethernet functional description: MAC

77.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

Transmit path

Table 805: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 805. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.

Receive path

Table 806: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the *VLAN tag Control register (ETH_MACVTCR)*.

Table 806. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

77.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to *Section 77.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or

replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 807: VLAN insertion or replacement based on VLTI bit*.

Table 807. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 857: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

77.5.3 Queue/channel-based VLAN tag insertion on Tx

The peripheral supports channel/queue based VLAN tag insertion on all transmitted packets.

Accessing queue/channel specific VLAN tag registers

Queue/channel specific VLAN tag registers are accessed using indirect addressing through the VLAN inclusion register (*VLAN inclusion register (ETH_MACVIR)*). VLAN type and tag value can be independently programmed for each queue/channel.

Enabling queue/channel based VLAN tag insertion on Tx

To enable this feature, set the CBTI bitfield in the VLAN inclusion register (*VLAN inclusion register (ETH_MACVIR)*). The VLAN tag is then inserted on every packet that is transmitted from a queue/channel, with the programmed tag value taken from queue/channel specific VLAN tag register

Programming guidelines for channel based VLAN tag insertion

See *Section 77.9.19: Programming sequence for queue/channel- based VLAN inclusion register on page 4461*.

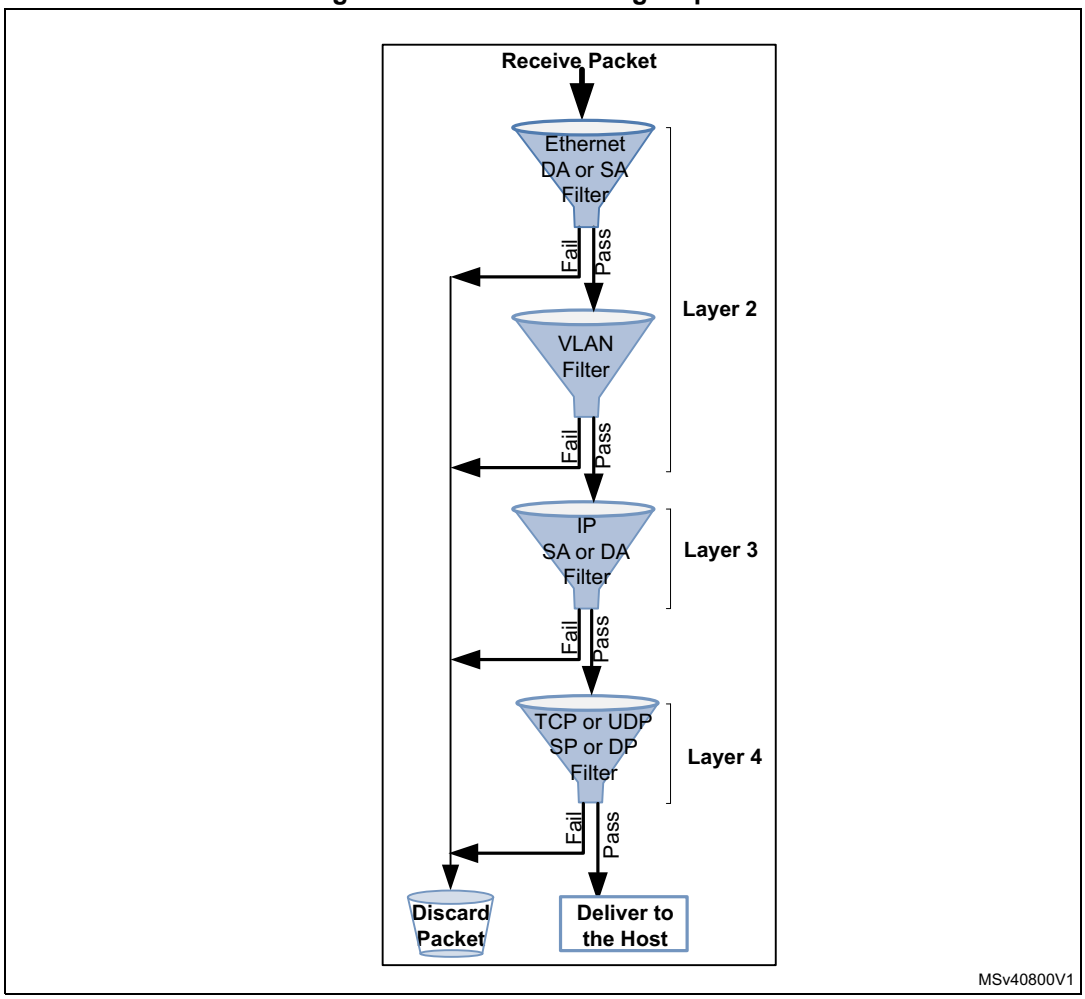
77.5.4 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 959* shows the filtering sequence for Rx packets.

Figure 959. Packet filtering sequence



The sequence shown in *Figure 959* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of *Packet filtering control register (ETH_MACPFR)* control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in *MAC Address x high register (ETH_MACAxHR)*. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of *MAC Address x high register (ETH_MACAxHR)*.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see *Table 809*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 808 and *Table 809* summarize the DA and SA filtering based on the type of packets received.

Note: When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 808. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
Multicast	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 809. Source address filtering

Packet type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 77.9.17: Programming guidelines to perform VLAN filtering on the receiver* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag Control register (ETH_MACVTCR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of *VLAN tag Control register (ETH_MACVTCR)*. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the *VLAN tag Control register (ETH_MACVTCR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag Control register (ETH_MACVTCR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse

match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 810 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 810*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag Control register (ETH_MACVTCR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in *VLAN tag Control register (ETH_MACVTCR)*.

Table 810. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM Bit	VLAN Hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, ‘X’ represents any value.

Extended receive VLAN filtering and routing

The MAC receiver can classify the received packets based on VLAN tag, and steer them to a specific Rx DMA channel. The MAC compares the VLAN tag of the received frame with all the enabled and relevant filters, and provides a filtering result. If any of the perfect filters give a pass result and if the respective filter DMA channel number is enabled, the frame is routed to that DMA channel.

In addition to filtering, routing can also be done. For more details about routing, see Extended VLAN-based DMA selection in *Section 77.4.4: Multiple channels and queues support*.

Comparison modes

For each VLAN tag filter, the application has the following comparison options:

- It can program the MAC to compare an outer VLAN tag or an inner VLAN tag with the programmed VID.
- It can choose if 12 or 16 bits of the VID field need to be compared.
- Type check can be disabled or enabled for each filter. If enabled, the application can choose if the VID comparison is for SVLAN or CVLAN type frames only.

  For example, if a filter is enabled for 16-bit comparison, SVLAN type, and Outer VLAN tag, any single or double VLAN-tagged frames with Outer SVLAN tags are compared with this filter, and a pass or fail result is obtained.

Note: The inner VLAN tag comparison is applicable only if Double VLAN tag processing is enabled in VLAN tag Control register (ETH_MACVTCR).

Filtering

When the Extended RX VLAN filtering and routing feature is enabled, the application can enable both perfect and Hash filtering. The overall VLAN filter result is based on the perfect filter result and the Hash filter result (if enabled). The filter result is passed to the application as part of the status bits.

Perfect filtering is done based on the MAC VLAN tag filters defined in *VLAN tag data register (ETH_MACVTDR)*. For each VLAN tag filter, the MAC compares the relevant VLAN tag ID and provides a result. If any one of the VLAN tag filters gives a match, the frame is considered to have passed the VLAN tag filters. If the frame mismatches all the filters, the frame is considered to have failed the VLAN filter. This behavior is applicable only when the inverse filtering is not enabled in the *VLAN tag Control register (ETH_MACVTCR)*.

If inverse filtering is enabled and the frame has mismatched all the relevant filters, then it is considered to have passed the VLAN filter. If the frame matches any one of the relevant filters, then it is considered as a fail. If none of the enabled filters can perform a comparison or if none of the filters are enabled, then the frame is bypassed to the application.

The overall filter result and the values programmed in the VTFE and RA bitfields of the *Packet filtering control register (ETH_MACPFR)* determine if the frame must be dropped or forwarded to the application:

- When RA = 1 or VTFE = 0

  The MAC forwards the frame, irrespective of the filter result.
- When RA = 0 and VTFE = 1

  The MAC forwards the frame only if the VLAN tag filter status is pass. After forwarding a frame to the application, the relevant filter result is indicated through the status bits.

Programming guidelines for Extended VLAN filtering and routing on reception

See *Section 77.9.18: Programming guidelines for extended VLAN filtering and routing on reception on page 4461*.

VLAN filter status

The Extended receive VLAN filtering and routing feature provides two status bits to indicate the comparison result of the VLAN tags.

By default, the MAC indicates the VLAN filter status through one bit in the status (VF) in RDES2. When the Extended receive VLAN filtering and routing is enabled, two status bits are used to indicate the comparison result of VLAN tags. The Outer VLAN tag filter pass and

Inner VLAN tag filter pass bits are defined in the following positions in various configurations. The status indicated through these bits is highly dependent on how the fields are programmed:

- In RDES2:
  - Bit 15: Outer VLAN tag filter status
  - Bit 14: Inner VLAN tag filter status
- Outer VLAN tag filter status (OTS)
  - In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame Outer VLAN tag has matched one of the VLAN tag filters.
  - If this bit is reset, it indicates that the frame Outer VLAN tag has either failed the relevant Outer VLAN tag filters or bypassed them.
  - If none of the filters are enabled for Outer VLAN tag comparison, then this bit is reset.
  - If inverse filtering is enabled and this bit is set, then the frame VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of the filters or bypassed all the filters programmed for Outer VLAN tag comparison.
  - This bit is valid for both Single and Double VLAN-tagged frames.
- Inner VLAN Tag Filter Status (ITS)
  - In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame Inner VLAN tag has matched one of the VLAN tag filters.
  - If this bit is reset, it indicates that the frame Inner VLAN tag has either failed the relevant Inner VLAN tag filters or bypassed them.
  - If none of the filters are enabled for Inner VLAN tag comparison, then this bit is reset.
  - If inverse filtering is enabled and this bit is set, then the frame VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of the filters or bypassed all the filters programmed for Inner VLAN tag comparison.
  - This bit is valid for only Double VLAN-tagged frames, when Double VLAN processing is enabled.

The application must monitor the status bits and the programming to determine if the frame has passed or failed the VLAN filter.

Table 811 and *Table 812* show the possible filter combinations and the corresponding filter results. These tables explain the scenarios when Double VLAN processing and Hash VLAN filter are enabled.

Table 811 shows the possible values of OTS and ITS status bits when at least one perfect filter is enabled.

Legend for *Table 811*:

- VTIM - VLAN Tag Inverse Match Enable: bit 17 of *VLAN tag Control register (ETH_MACVTCR)*.
- HFO - Hash Filter enabled for Outer VLAN Tag Comparison: bit 26 and bit 27 of *VLAN tag Control register (ETH_MACVTCR)*.
- HFI - Hash Filter enabled for Inner VLAN Tag Comparison): bit 26 and bit 27 of *VLAN tag Control register (ETH_MACVTCR)*.
- PFO - Perfect filter comparison enabled for Outer VLAN Tag: any of the MAC VLAN tag filters in *VLAN tag data register (ETH_MACVTDR)* is enabled (bit 16 is set) and programmed for Outer VLAN tag comparison (bit 20 is set to 0).
- PFI - Perfect Filter comparison enabled for Inner VLAN Tag: any of the MAC VLAN tag filters in *VLAN tag data register (ETH_MACVTDR)* is enabled (bit 16 is set) and programmed for Inner VLAN tag comparison (bit 20 is set to 1).
- OTS - Outer VLAN tag filter status
- ITS - Inner VLAN tag filter status

Table 811. OTS and ITS bit values with at least one perfect filter enabled

VTIM	HFO	HFI	PFO	PFI	OTS	ITS
0	0	0	0	1	0	1/0
0	0	0	1	0	1/0	0
0	0	0	1	1	1/0	1/0
0	1	0	1	1	1/0	1/0
0	1	0	1	0	1/0	0
0	1	0	0	1	1/0	1/0
0	0	1	1	1	1/0	1/0
0	0	1	1	0	1/0	1/0
0	0	1	0	1	0	1/0
1	0	0	0	1	0	1/0
1	0	0	1	0	1/0	0
1	0	0	1	1	1/0	1/0
1	1	0	1	1	1/0	1/0
1	1	0	1	0	1/0	0
1	1	0	0	1	1/0	1/0
1	0	1	1	1	1/0	1/0
1	0	1	1	0	1/0	1/0
1	0	1	0	1	0	1/0

Table 812 shows the possible values of OTS and ITS status bits when none of the perfect filters are enabled and only the VLAN Hash filter is enabled.

Table 812. OTS and ITS bit values with only VLAN Hash filter enabled

VTIM	HFO	HFI	OTS	ITS
0	0	0	0	0
0	1	0	1/0	0
0	0	1	1/0	1/0
1	0	0	1/0	0
1	1	0	1/0	0
1	0	1	1/0	1/0

- When no perfect filters are enabled, any VLAN packet is considered to have bypassed the perfect filter.
- When VLAN Hash filter is enabled for one of the tags, the respective status bit depends on the result of the filter. Status bits are set to 0 when VLAN Hash filter is not enabled.
- The value 1/0 for the ITS/OTS field indicates that the final result depends on the result of the enabled relevant filter.

Example 1

The second row of *Table 812* indicates that at least one perfect filter is enabled for Outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN filtering is not enabled. The bit OTS is given as 1/0. If the received frame passes at the least one of the enabled Outer VLAN tag filters, then the bit is set to 1. If the frame does not pass any of the enabled Outer VLAN tag filters, then the bit is set to 0.

Example 2

The last row of *Table 812* indicates that inverse filtering is enabled, Hash filter and at least one perfect filter is enabled for Inner VLAN tag comparison. If the received frame Inner VLAN tag mismatches with both the Hash filter and all the enabled perfect filters, then the frame has the ITS bit set to 1. Otherwise, the bit is set to 0. OTS bit is set to 0 as comparison is not performed.

VLAN stripping

Each of the VLAN tags has individual control over stripping. The programming options of always strip, never strip, strip on pass and strip on fail are available. Inner or Outer VLAN tag stripping is based on the pass or fail results of the individual tag. If a tag is bypassed by all the relevant filters, stripping is not applicable for the tag.

- If strip-on-pass is enabled for the outer VLAN tag, stripping is done only if the Outer VLAN tag has passed the relevant filters. The Outer VLAN tag filter result bit is set.
- If strip-on-fail is enabled for the outer VLAN tag, stripping is done only if the Outer VLAN tag has failed the relevant filters. The Outer VLAN Tag Filter Result Bit is reset.
- If the Outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), the tag is not stripped, though the status bit is still 0.
- As multiple filters are enabled, it is possible that the received VLAN frame matches more than one filters. The VLAN tag value is not always deterministic from the filter status bits.
- If the application strips the VLAN tag based on the filter result, it might lose the VID. So, if stripping is enabled for any of the tags, the tag can be put in the status. The

application must enable the respective "VLAN Tag in Status" bit[24] or bit[31] in the *VLAN tag Control register (ETH_MACVTCR)*.

VLAN filter fail packet queue

When VLAN filtering is enabled, the VLAN filter fail packets can be routed to a programmable queue (VFFQ) when RA = 1 or VTFE = 0 and the enable bit (VFFQE) for the queue is set:

- RA and VTFE bits are implemented in *Packet filtering control register (ETH_MACPFR)*.
- VFFQ and VFFQE fields are implemented in *Rx Queue control register (ETH_MACRXQCR)*.

The packets that pass the VLAN filtering are routed based on the VLAN tag priority field. The VLAN tag priorities can be assigned to Rx queues by programming the PSRQ bitfield in *Rx queue control 2 register (ETH_MACRXQC2R)*. The packets that fail the VLAN filter are discarded if RA = 0 or VTFE = 1. However when RA = 1 or VTFE = 0, the VLAN filter fail packets are still forwarded to the application. In such a scenario, when the VLAN Filter Fail Queue Enable (VFFQE) bit is set, the VLAN filter fail packets are forwarded to the Rx queue number programmed in the VFFQ bit. If VFFQE = 0, the Rx queue number is determined by the VLAN priority mapping as per the PSRQ bitfield.

Table 813 shows the Rx queue routing table for Unicast tagged packets, with DA/SA filter enabled.

Table 813. Rx Queue routing table for Unicast-tagged packets(1)

RA	VFTE	SA/DA filter result	VLAN filter result	VFFQE	Queue routing
X	X	PASS	PASS	X	PSRQ
0	0	PASS	FAIL	0	PSRQ
0	0	PASS	FAIL	1	VFFQ
0	X	FAIL	X	X	DROPPED
0	1	PASS	FAIL	X	DROPPED
1	X	FAIL	X	0	UFFQ(2)/PSRQ
1	X	FAIL	X	1	UFFQ(2)/VFFQ
1	X	PASS	FAIL	0	PSRQ
1	X	PASS	FAIL	1	VFFQ

1. X = Don't care condition.
2. When UFFQE is enabled, else PSRQ.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

  – All enabled Layer 3 and Layer 4 fields match.

  – At least one of the enabled field matches and other fields are bypassed or disabled

    When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 Address filter 0 register (ETH_MACL4A0R)*
- *Layer3 Address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 Address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 Address 2 filter 0 register (ETH_MACL3A20R)*
- *Layer3 Address 3 filter 0 register (ETH_MACL3A30R)*

The second, and independent set of registers are: *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, *Layer 4 address filter 1 register (ETH_MACL4A1R)*, *Layer3 address 0 filter 1 Register (ETH_MACL3A01R)*, *Layer3 address 1 filter 1 register (ETH_MACL3A11R)*, *Layer3 address 2 filter 1 Register (ETH_MACL3A21R)* and *Layer3 address 3 filter 1 register (ETH_MACL3A31R)*.

77.5.5 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- *IEEE 1588 timestamp support*
- *IEEE 1588 system time source*
- *IEEE 1588 auxiliary snapshots*
- *Flexible pulse-per-second output*

- *PTP timestamp offload function*
- *One-step timestamp*"

IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

  – Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control Register (ETH_MACTSCR)*.

  – Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 814. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

  For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control Register (ETH_MACTSCR)*.

- Boundary clock

The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

– The clock data sets are common to all ports of the boundary clock.

– The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in *Table 815*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control Register (ETH_MACTSCR)*.

Table 815. End-to-end transparent clock: PTP messages for snapshot

PTP messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Table 823*.

Table 816. Peer-to-peer transparent clock: PTP messages for snapshot

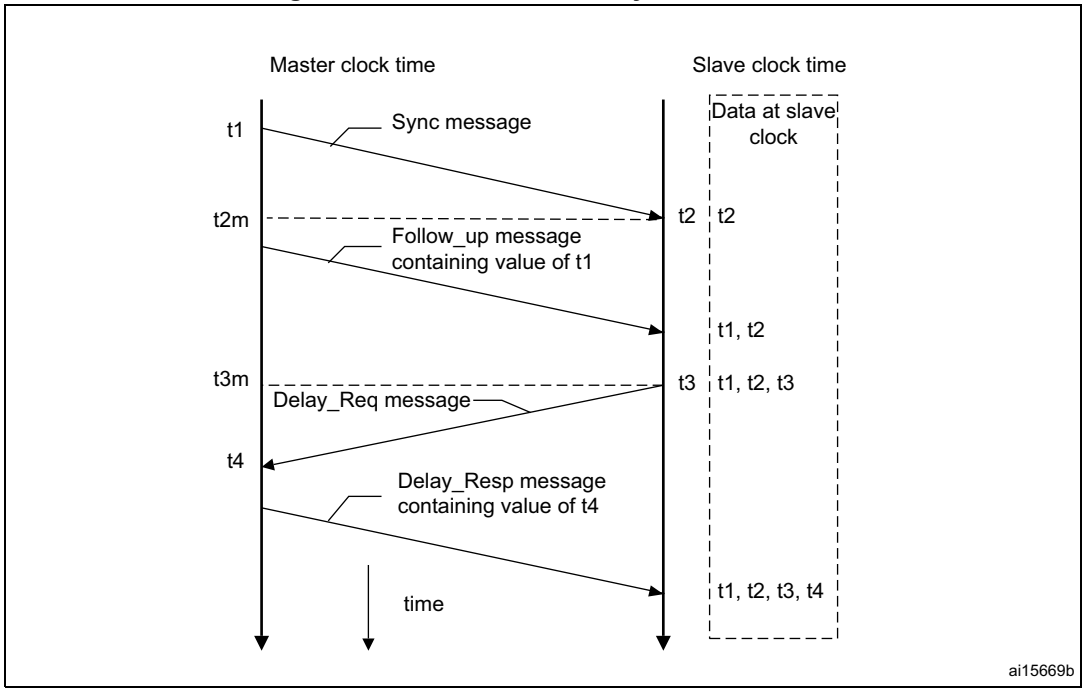
PTP messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control Register (ETH_MACTSCR)*.

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 960* shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

Figure 960. Networked time synchronization



As shown in *Figure 960*, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at GMII or MII.
2. The slave receives the SYNC message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the GMII or MII interface.

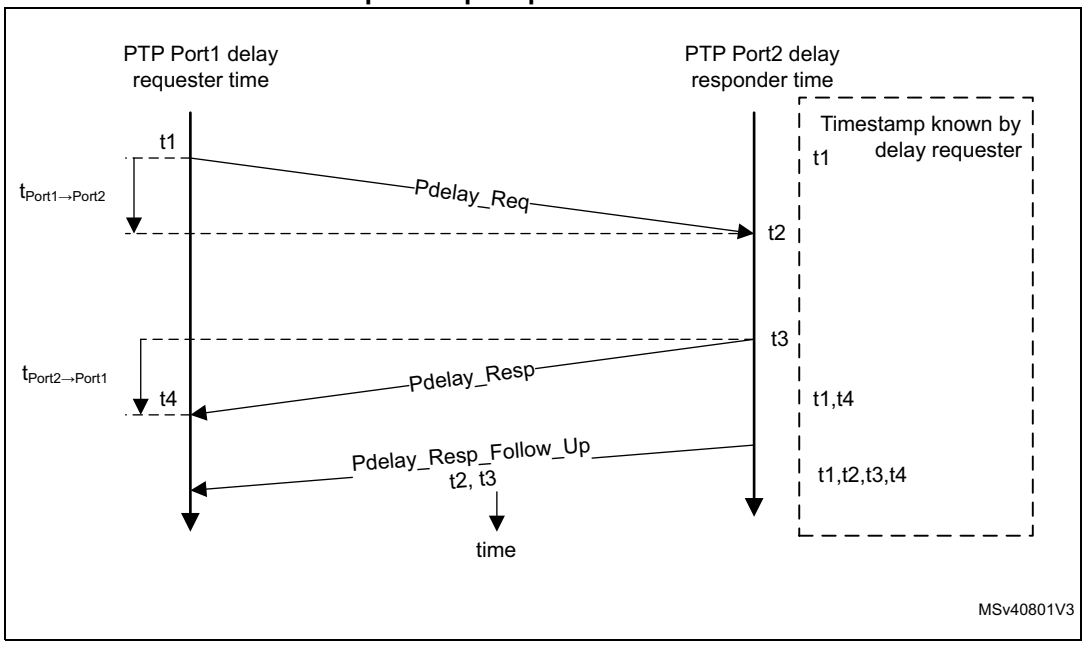
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the GMII or MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 961* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 961. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 961*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.

3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   – Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp correction

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (GMII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the GMII and PTP clocks.

Ingress correction

In the Receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the input of the core to the internal capture point

   The internal ingress latency can be read from the *Timestamp Ingress Latency register (ETH_MACTSILR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed.

Therefore, the software must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).

The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

Note: The value written to the register must be negative (two's complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with "10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is –5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of ~0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

Egress correction

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the internal capture point and the output of the core

   This internal egress latency can be read from the *Timestamp Egress Latency register (ETH_MACTSELR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface and the operating speed. Therefore, the software must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization error

   The timestamp correction because of synchronization is compensated by adding

$$\text{EGRESS_SYNC_CORR} = (1 \times \text{PTP_CLK_PER} + 4 \times \text{TX_CLK_PER})$$

Table 817 lists the egress and ingress latency values for various PHY interfaces:

Table 817. Egress and ingress latency for PHY interfaces

PHY interface		Egress latency	Ingress latency
RGMII	1 Gbps	12	12
RGMII	100 Mbps	40	40
RGMII	10 Mbps	400	400
RMII	100 Mbps	40	120
RMII	10 Mbps	400	800

Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of GMII or MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- Maximum PTP clock frequency

  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- Minimum PTP clock frequency

  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the GMII or MII clock domain. This relationship is given by the following equation:

  3 × PTP clock period + 4 × GMII/MII clock period ≤ Minimum gap between two SFDs

  The GMII or MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 818*.

Table 818. Minimum PTP clock frequency example

Mode	Minimum gap between two SFDs	Minimum PTP frequency with external timestamp Input	Minimum PTP frequency with internal timestamp
10 Mbps Full-duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	~45 KHz	5 MHz
10 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	170 KHz	5 MHz
100 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	~0.5 MHz	5 MHz

Table 818. Minimum PTP clock frequency example (continued)

Mode	Minimum gap between two SFDs	Minimum PTP frequency with external timestamp Input	Minimum PTP frequency with internal timestamp
100 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	4.55 MHz	5 MHz
1000 Mbps Full-duplex	84 GMII clocks (64 clocks for a 64-byte packet + 12 clocks of min IFG + 8 clocks of preamble)	~4.7 MHz	5 MHz
1000 Mbps Half-duplex	24 GMII clocks (4 clocks for a JAM pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	18.75 MHz	18.75 MHz

PTP processing and control

Table 819 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 819. Message format defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 820 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 819*.

Table 820. Message format defined in IEEE 1588-2008

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x0800	IPv4 datagram
IP version and header length	14	0x45	IP version is IPv4
Layer 4 protocol	23	0x11	UDP
IP multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	0x013F, 0x0140	0x013F: PTP event messages(1) 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

PTP frames over IPv6

Table 821 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 819*.

Table 821. IPv6-UDP PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer 4 protocol	20(1)	0x11	UDP
PTP multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP destination port	56, 57a	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP message type field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension header is not defined for PTP packets.

PTP packets over Ethernet

Table 822 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 822. Ethernet PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC destination multicast address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC packet type	12, 13	0x88F7	PTP Ethernet packet
PTP control field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of *Timestamp control Register (ETH_MACTSCR)* is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the GMII or MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the Transmit Descriptor (see *Section 77.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding

Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the *Timestamp control Register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 823 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in *Timestamp control Register (ETH_MACTSCR)*.

Table 823. Timestamp snapshot dependency on ETH_MACTSCR bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See *Section : System time correction* in *Section 77.9.10: Programming guidelines for IEEE 1588 timestamping on page 4448*.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral provides the following options for using the reference timing source in a node:

- External Timestamp Input

  This option takes an external 64-bit timing reference using the ptp_timestamp_i[63:0] bus.

  The 64-bit timing reference is split into the following two 32-bit signals:

  – Upper 32 bits providing the time in seconds

  – Lower 32 bits providing the time in nanoseconds

  This timing reference is used to timestamp the packets.

- Internal reference time (80 bits)

  This option takes only the reference clock input and uses it to internally generate the reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds field

  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.

- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:

  – **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.

  – **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.

    These modes can be set through TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)*.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the GMII.

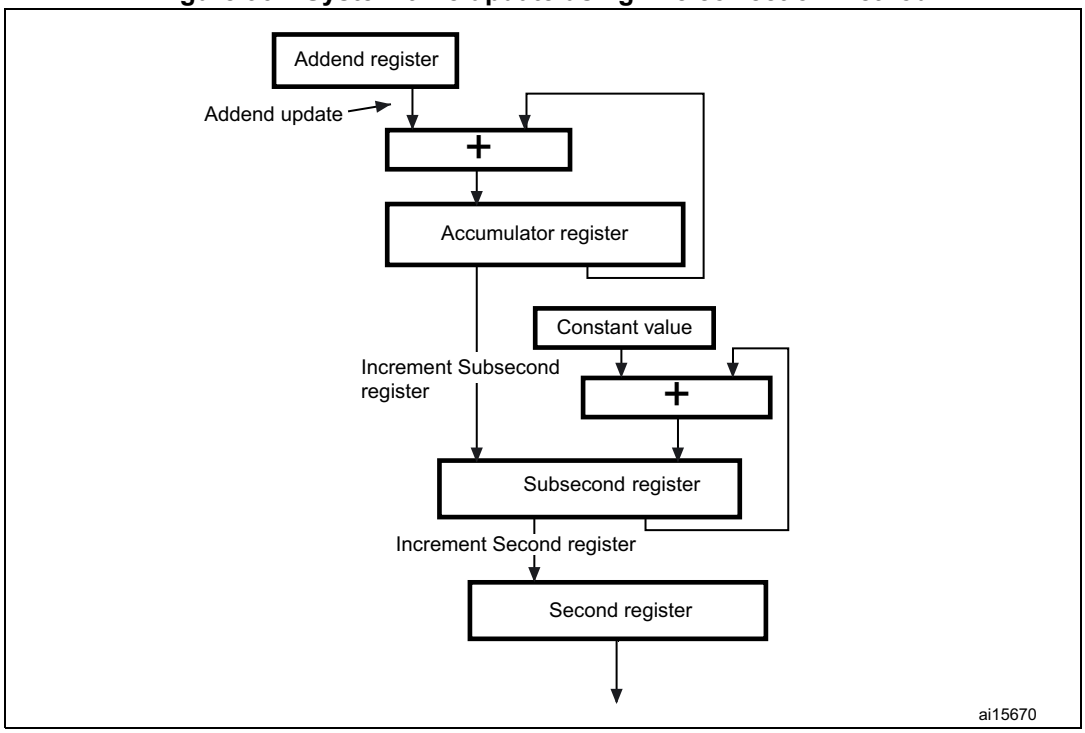
The system time counter can be initialized or corrected using either the coarse or the fine correction method.

In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 962*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 962*.

Figure 962. System time update using fine correction method



The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In *Figure 962*, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

Note: When External Time Update is enabled, the System Time module is not available.

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time MasterSyncTime_n the master sends the slave clock a SYNC message. The slave receives this message when its local clock is SlaveClockTime_n and computes MasterClockTime_n as follows:

$$\text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n$$

2. The master clock counts for current Sync cycle, $\text{MasterClockCount}_n$ is

$$\text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, SlaveClockCount_n is

$$\text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, ClockDiffCount_n is

$$\text{ClockDiffCount}_n = \text{MasterClockTime}_n - \text{SlaveClockTime}_n$$

5. The frequency-scaling factor for slave clock, FreqScaleFactor_n is

$$\text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n$$

6. The frequency compensation value for Addend register, $\text{FreqCompensationValue}_n$ is

$$\text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n \times \text{FreqCompensationValue}_{n-1}$$

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to *Section 77.9.10: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

IEEE 1588 auxiliary snapshots

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where x = 1 to 4) sideband signal.

This feature is independent of whether the system time is generated internally or given as input (on ptp_timestamp_i[63:0] bus).

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of *Timestamp status register (ETH_MACTSSR)*.

Flexible pulse-per-second output

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out1 outputs:

- Fixed pulse-per-second output

  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.

- Flexible pulse-per-second output

  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:

  The start and stop times are programmed through *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.

  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through *PPS x width register (ETH_MACPPSWxR)* and *PPS x interval register (ETH_MACPPSIxR)*, respectively.

Note: By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- *the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.*
- *PPS x target time seconds register (ETH_MACPPSTTSxR) and PPS x target time nanoseconds register (ETH_MACPPSTTNxR) are used only for generating target time reached interrupt; they are not used for PPS output generation.*
- *TRGTMODSEL0/1/2/3 must be programmed to 0.*
- *the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).*

Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Control of up to two independent PPS outputs, available externally (ETH_PPS_OUT) and internally (eth_ptp_pps_out1)
- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the *Subsecond increment register (ETH_MACSSIR)*. The pulse width can be programmed from 1 to 232–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control Register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 962).

Refer to *Section 77.9.14: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

Each flexible PPS output is associated to independent Target Time registers, *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)* indicates that the

synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.

Each flexible PPS output is associated to independent interval and width registers, *PPS x interval register (ETH_MACPPSIxR)* and *PPS x width register (ETH_MACPPSWxR)*.

PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through *PTP Offload control register (ETH_MACPOCR)*. 80-bit PTP node identity is configured through the following three registers: *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 824* indicates the PTP message generation criteria.

Table 824. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req

Table 824. PTP message generation criteria (continued)

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for PTP Offload feature.					

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP Offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC *only when the bit is reset.*

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission

of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in *Table 824*.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

PTP packet generation

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

Table 825. Common PTP message header fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5

Table 825. Common PTP message header fields (continued)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

PTP message header fields

- **messageType**
  The following encoded values are used for PTP message types:
  – SYNC: 0000
  – Delay_Req: 0001
  – Pdelay_Req: 0010
  – Pdelay_Resp: 0011
  – Delay_Resp: 1001
- **transportSpecific**
  The following transport protocol encoding is used:
  – Default PTP over Ethernet: 0000
  – 802.1AS mode: 0001
- **versionPTP**
  It is always set to 2 because PTP version 2 is supported.
- **domainNumber**
  This field contains the value from the *PTP Offload control register (ETH_MACPOCR)*.
- **flagField**
  The following values are used:
  – alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  – twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  – unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address
- **correctionField**
  For more information, see *Table 826*.
- **sourcePortIdentity**
  This field takes the value programmed in the *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.
- **sequenceId**

Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.

- **controlField**

  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011
- **logMessageInterval**
  - SYNC:

    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.
  - Delay_Resp:

    This field contains the sum of DRSYNCR and logSyncInterval value taken from the *Log message interval register (ETH_MACLMIR)* for a multicast PTP message and 0111 1111 for unicast PTP message.
  - Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111

    where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- 2^{N} seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 2^{1}; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2^{-1} = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2^{-30} = 1 ns.
- When logSyncInterval is programmed to –-5, the interval is 2^{-5} = 33.55 ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**

  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type.

  For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.
- **originTimestamp**

  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.
- **receiveTimestamp**

  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.
- **requestingPortIdentity**

  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.
- **requestReceiptTimestamp**

  For the Pdelay_Resp PTP message, this field is set to 0.

One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 826 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Timestamp control Register (ETH_MACTSCR)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 826. MAC Transmit PTP mode and one-step timestamping operation

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)

Table 826. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
							Delay_Req (correction field for residence time and egress asymmetric correction)
00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
01	X	0	End-to-end Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Pdelay_Req (correction field for residence time and egress asymmetric correction)
						Ingress TS	Pdelay_Resp (correction field for residence time and ingress asymmetric correction)
01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

Table 826. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Delay_Req (correction field for residence time and egress asymmetric correction)
11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

77.5.6 Time sensitive networking (TSN)

This section describes how the Ethernet peripheral core supports time sensitive network (TSN) features.

TSN is a set of standards, developed by the IEEE 802.1 work group, which define the mechanisms for the time-sensitive transmission of data over Ethernet.

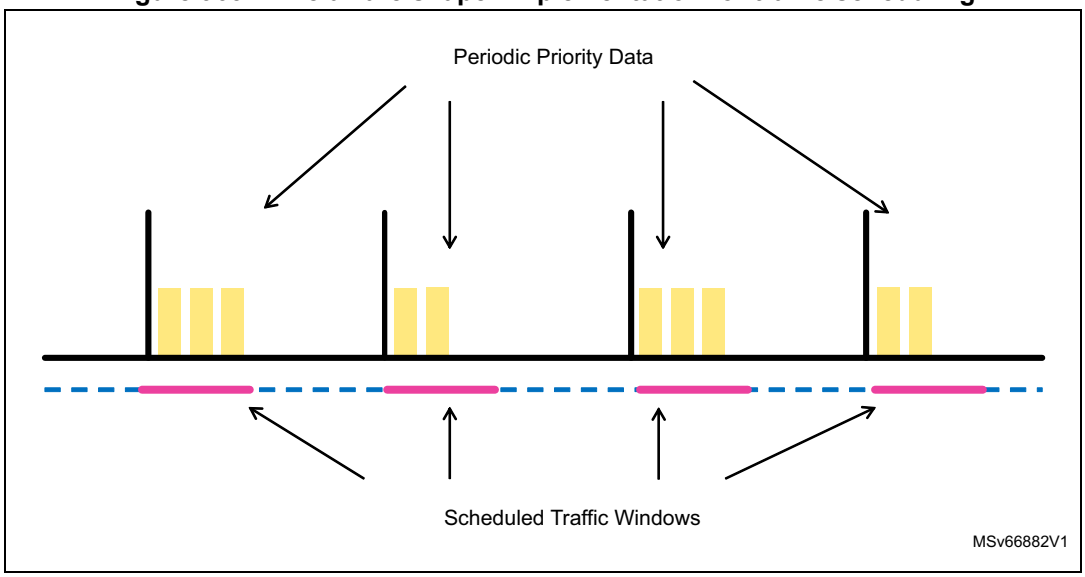
The present section describes:

- *Enhancements for scheduled traffic (EST)*
- *Frame preemption (FPE)*
- *Time-based scheduling (TBS)*

Enhancements for scheduled traffic (EST)

The IEEE 802.1Qbv-2015 defines the schedule for each queue on every egress port that makes the peripheral aware of the traffic arrival schedule. This information can be used to block lower priority traffic from being transmitted in this time window/slot. This ensures that scheduled traffic is forwarded from sender to receiver through all the network nodes with a deterministic delay.

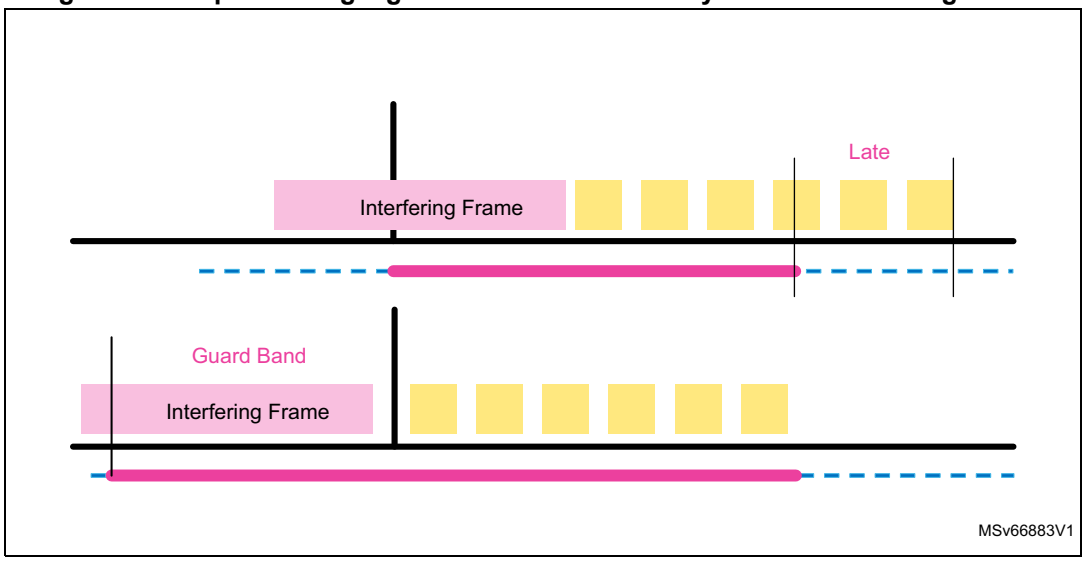
Figure 963. Time aware shaper implementation for traffic scheduling



One of the important requirements to achieve a low latency is to make sure there are no interfering frames during the scheduled windows that are reserved to high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

As shown in *Figure 964*, if an interfering frame transmission begins just before the start of a reserved time period, the transmission can be extended into the reserved window and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

Figure 964. Implementing a guard band to avoid delays due to interfering frames



A larger guard band equates to a less efficient use of network bandwidth. However, with the implementation of IEEE802.1Qbu (frame preemption), this issue is addressed. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

During the guard band, only the frames for which the entire frame can be complete before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

The transmit scheduling embeds the following feature to be able to support EST:

- Gate control list (GCL)
- Enforced gate controls in the scheduler
- Gate open duty cycle taken into account in the computation of the idleSlope (CBS)

Gate control list implementation (GCL)

Figure 965 shows the block diagram extracted from the IEEE 802.1Qbv specifications that illustrates how the gate control list governs the gate close (C) and open (O) events based on the schedule provided for each event.

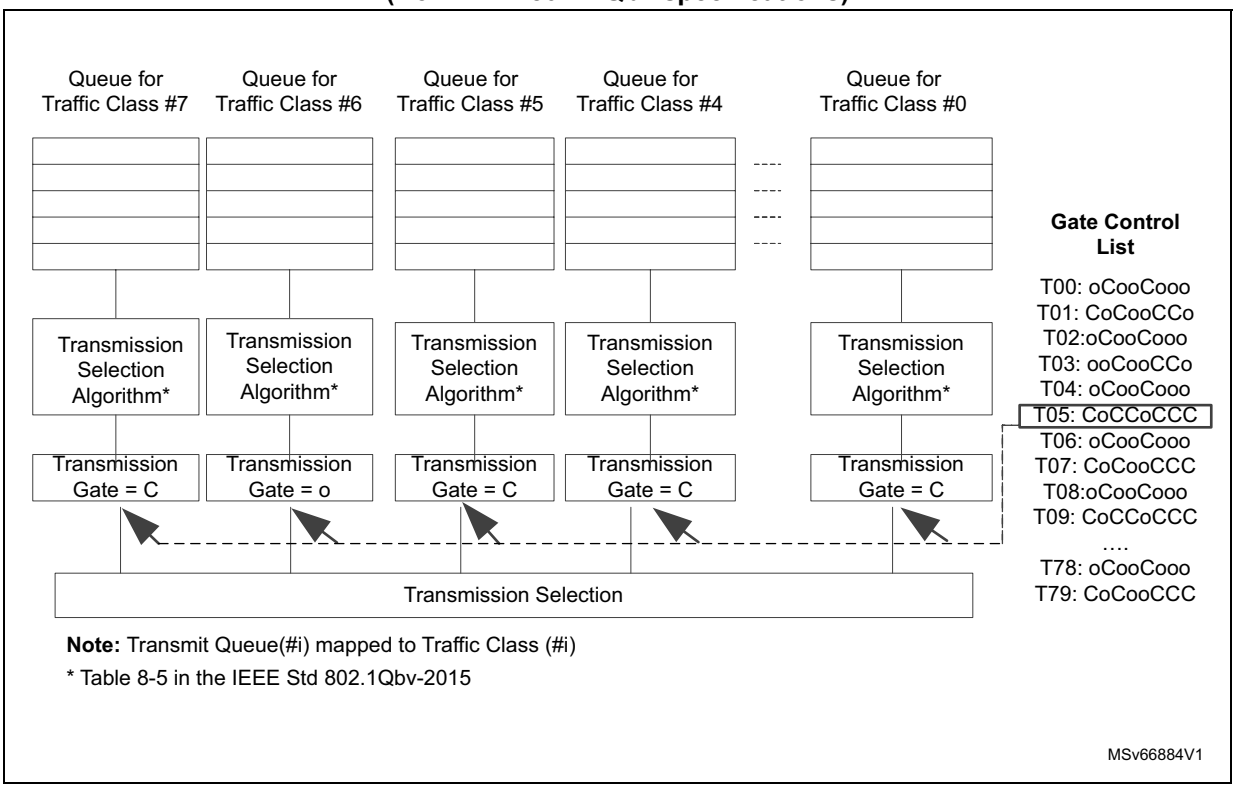
The GCL consists of two parts:

- Time interval

  It defines the time (expressed in ns) during which the gate controls are valid and should be applied before reading the next gate controls from the list. The maximum time interval is 16 ms. The left shift of the time interval up to 7 bits is supported to be able to apply a multiplication factor ranging from 1 to 128 ns (in steps of 2^n). The maximum value (post shifting) of this field must be 999,999,999 ns.
- Gate control

  Defines the Open (O) represented by logic 1 or Close (C) represented by logic 0 state for the gate of each traffic class (TC).

Figure 965. GCL governing gate close and open events block diagram (from IEEE 802.1 Qbv specifications)



The implementation of GCL consists of the following two gate-control lists:

- Hardware owned list (HWOL), which can be accessed only by hardware
- Software owned list (SWOL), which can be accessed only by software

The access to these lists is mutually exclusive. The ownership of the list is set by hardware in the SWOL field of *EST Status register (ETH_MTLESTSR)*. *Table 827* shows how the GCL is implemented.

Table 827. External memory used for holding the two gate-control lists

Gate control (up to 8 bits)	Time interval in ns	Gate-control list
OOCCCCCC	T0	HWOL or SWOL
OOOOCCCC	T1	
CCCCOOCC	T2	
CCCCCCOO	T3	
I	I	
I	I	
OCOCOCOO		
OOOOCCCC	Last T	

Table 827. External memory used for holding the two gate-control lists (continued)

Gate control (up to 8 bits)	Time interval in ns	Gate-control list
OOOOCCCC	T0	SWOL or HWOL
OOOOCCCC	T1	
OOCCCCCC	T2	
OOOOCCCC	T3	
I	I	
I	I	
OCOCCCCC		
OOCCCCCC	Last T	

Four registers are related to the gate control lists (one for each GCL). These registers are implemented through Indirect addressing using *EST Gate Control List register (ETH_MTLESTGCLCR)* and *EST Gate Control List Data register (ETH_MTLESTGCLDR)*.

- **64-bit base time register (BTR)**

  The base time register is a 64-bit register that holds the start time to execute the GCL. The format of the BTR is the same as the PTP format (upper 32-bit holds time in seconds and lower 32-bit hold time in ns).

  Once the execution of a given list starts, the peripheral can update the BTR value to indicate the next list execution start time.

- **40-bit cycle time (CTR)**

  The cycle time register is a 40-bit register that holds the time at which the execution of the GCL should be repeated. The cycle time register consists of an 8-bit value in seconds and a 32-bit value in ns (similar to the PTP time format with truncated second register).

  For a given GCL, the start time is given by the below formula:

  $$\text{Start time} = \text{Base time} + N \times \text{Cycle time}$$

  where N is an integer value representing the iteration number starting from 0 for the first iteration.

  If the GCL execution takes longer than the cycle time, then the list is truncated at the cycle time and the subsequent loop starts at the cycle time.

- **32-bit time extension (TER)**

  The time extension register (TER) is a 32-bit register that holds the amount of time (in ns) of which current GCL can be extended before switching to the new GCL. This is useful to avoid small fragments of the current list before switching to a new list. The time extension value is coded on 31 bits and stored in a 32-bit register, with the most significant bit forced to 0.

- **7-bit [gate control] list length (LLR)** (where n depends on the configured GCL depth)

  The list length register (LLR) is a 7-bit register that holds the integer value of the GCL length, which corresponds to the number of valid rows in each GCL. The processing of the GCL stops after the number of rows read is equal to the LLR value.

Transmission gating

A bridge or an end station can be used to allow transmission from each TC that is yet to be scheduled relative to a known timescale. To achieve this, a transmission gate is associated to each TC. The state of the transmission gate determines if the queued frames can be selected for transmission.

The state of a TC transmission gate can be in one of the following:

- Open

  Queued frames are selected for transmission according to the definition of the transmission selection algorithm associated to the TC.
- Closed

  Queued frames are not selected for transmission.

A frame on a traffic class queue is not available for transmission if the transmission gate is in closed state or if there is insufficient time available for transmitting the whole frame before the next gate-close event associated to this queue.

The peripheral has visibility on the current schedule of gate controls and the immediate next schedule of the gate controls. As a result, the maximum gate open period does not exceed the sum of the two time intervals since a frame is selected for transmission only if the gate is currently open and the duration of gate open interval is greater than the time taken to transfer the entire frame.

The implementation must know the frame size before the transmission, so that transmission overruns can be avoided and only the frames that can complete are scheduled. This can be achieved by:

- programming the OVHD field ([5:0] bits) of *EST Extended Control register (ETH_MTLESTECR)* with fixed overhead of the IPG or EIPG, preamble, and scheduler delay
- automatically predicting the variable overhead due to the offloads enabled for a packet in the MAC transmitter.

Consider the following while programming the EST related fields to ensure correct packet scheduling:

- Speed: 1 Gbps/100 Mbps
- PTOV (in ns): 9 PTP clock cycles
- CTOV (in ns): 3 PTP + 3 Transmit clock cycles
- The slot interval per packet (in ns) is given by the following formula:

$$\text{Slot interval per packet} = (\text{Packet size} + \text{Overhead} + \text{IPG or EIPG} + \text{preamble}) \times \text{time to transmit 1 byte}$$

The overhead and scheduler delays in bytes are equal to 17 Tx + (10 + X) application clock cycles, where X = 2 if VLAN tag insertion is enabled, X = 0 otherwise.

The peripheral adequately compensates for the data transfer delays from the buffer to the line by offsetting the current time with all the relevant delays (provided by the CTOV field of *EST Control register (ETH_MTLESTCR)*). This ensures that the provided schedule is always accurately implemented at the line.

Note: An overhead is added to the packet size to take into account the scheduler delay, IPG or EIPG, and preamble overheads on the line. The gate open window should therefore be able to accommodate at least 128 + overhead bytes to transmit a 128-byte frame.

In 100-Mbps mode of operation, the rounding down error is about 0.4 %. For example, the gate open duration should be at least 1035 bytes for transmitting 1000-byte frames at a speed of 100 Mbps (31 bytes for scheduler delay and 4 bytes for rounding down error).

The minimum time interval for GCL entry is the following:

$$\text{Min time interval for GCL entry} = \text{Time to transmit fragment size} + \text{Scheduler delay} + \text{IPG/EIPG} + \text{Preamble overheads}$$

Idle slope computation updates

When EST is enabled, the credit is accumulated only when the gate is open. The effective data rate of the idleSlope must therefore be increased to reflect the duty cycle for the transmission gate associated to the queue.

The idle slope (idleSlope) is computed based on the gate open time (GateOpenTime) and operating cycle time (OperCycleTime) values. Program the idleSlope registers (one per CBS enabled TC) based on the following equation:

$$\text{idleSlope} = (\text{operIdleSlope(N)} \times \text{OperCycleTime} / \text{GateOpenTime})$$

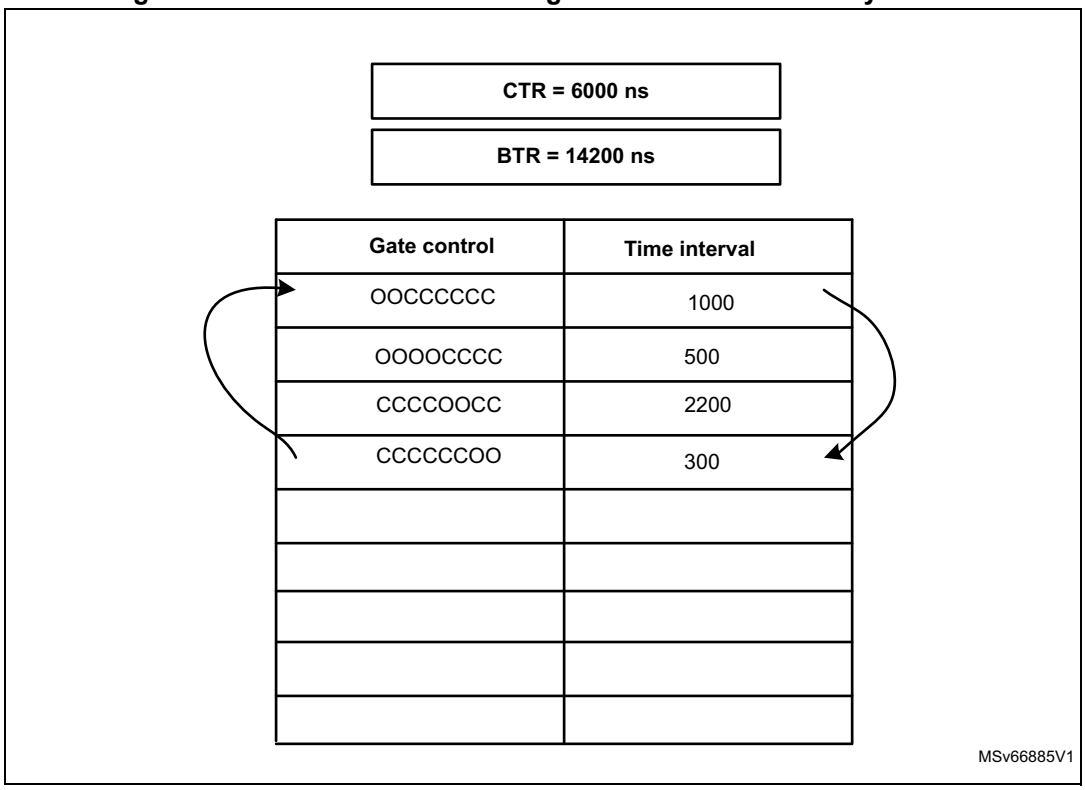
Operational details of GCL

Set the switch to software list through the SSWL bitfield in *EST Control register (ETH_MTLESTCR)*, so that the hardware can access the programmed gate control list. The first set of gate controls is applied when the current time is equal to the value in the base time register (BTR) and is held until the programmed time interval value.

To avoid transmission overruns, an additional gate control event is always read ahead from the list. This enables the GCL to determine the next gate close events (if any) for the TCs that are open.

The scheduling is based only on the gate open state and time interval of the current and subsequent schedule. An internal accumulator is used to add the time intervals when gate controls are applied. The sum of BTR + accumulator holds the time at which the next set of gate controls are planned to be applied.

Figure 966. GCL and associated registers - BaseTime and CycleTime



The GCL is read progressively from the first row adhering to the schedule. The read operations continue until the list length (defined in the LLR register) is reached. The execution of the list restarts at BTR + CTR time. At this point the BTR is incremented by the value in CTR to mark the beginning of a new cycle. In the absence of gate controls, all the gates are deemed to be in open state during the execution of the list.

In cases where the list execution time is greater than the CycleTime, the list is truncated and the next iteration starts when the current time equals BTR + CTR.

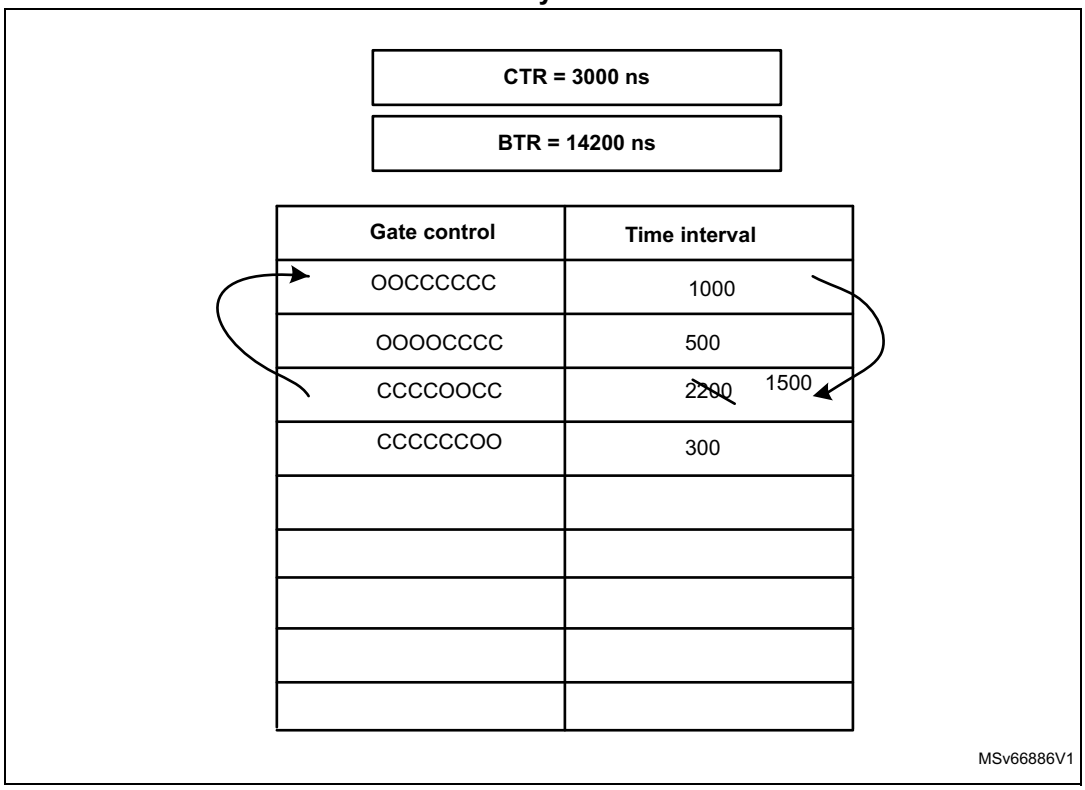
Table 828. GCL and associated registers - BTR and CTR

Current time	Gate control applied	Accumulator value	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17900	CCCCCCOO	4000	14200
18200	OOOOOOOO	0	20200
20200	OOCCCCCC	1000	20200
21200	OOOOCCCC	1500	20200

In the example given in *Table 828*, the execution starts at 14200 and the first set of gate controls, OOCCCCCC is immediately applied. Since the time interval is 1000 ns, the next set of gate controls is applied at 14200 (BTR) + 1000 (accumulator) = 15200 ns as shown in *Table 828*. Since there are no gate controls available after the execution of the last gate

control and before the next iteration of the loop, the gates are deemed to be in open state during that time period as shown at time 18200 in *Table 828*.

Figure 967. GCL and associated registers - BaseTime and CycleTime list execution time > CycleTime



Since the list execution takes longer than the allocated CycleTime, the list is truncated and the list is started from the BTR+CTR as shown in *Table 829*.

Table 829. GCL and associated registers - BTR and CTR, execution time > CycleTime

Current time	Gate control applied	Accumulator value	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	OOCCCCCC	1000	17200
18200	OOOOCCCC	1500	17200
18700	CCCCOOCC	3700	17200

While applying the third set of gate controls, BTR + CycleTime (17200 ns) is lower than BTR + accumulator (17900), so the list is truncated and execution switches to a new iteration at 17200.

Installing a new GCL

When a new software-programmed GCL is available and must be executed at the new BTR value, the switch to the new GCL can be done in one of the following ways:

- New base time aligned with current schedule
- New base time unaligned with current schedule

New base time aligned with current schedule

If the cycle time value for the new gating cycle is the same as the cycle time of the current gating cycle, and if the BTR chosen for the new gating cycle (new BTR) is an integer multiple of the current cycle time (+ current BTR), then the new gating cycle exactly lines up with the old gating cycle, that is, the cycle start times for the new gating cycle are the same as they would have been for the old configuration. This can be considered as the ideal case and enables the new gating cycle to be installed and executed with no timing issues. The peripheral completes the execution of an iteration of the current list and switches to the new list at the beginning of the BaseTime listed in the new list.

If (New BaseTime ≥ Current time) ConfigChangeTime = New BaseTime

Else If (New BaseTime < Current Time)

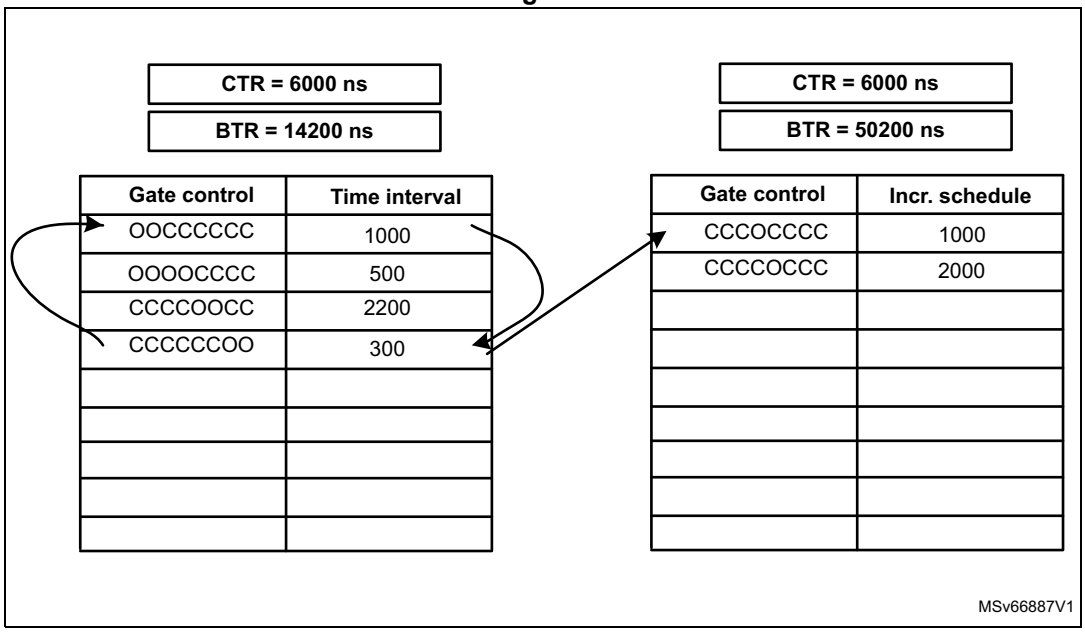
1. Set the BTRError (BTRE bit in ETHMTLESTR)
2. ConfigChangeTime = New BaseTime + N × New CycleTime)

where N is the smallest integer for which the relation ConfigChangeTime ≥ CurrentTime and (N ≤ 8) is TRUE.

When N is higher than 8, the hardware cannot auto-recover and the loop count value in BTR error loop count (BTRL bitfield in ETH_MTLESTSR) is set to 0b1111 1111 requiring the software to reprogram the new base time.

Figure 968 illustrates the installation of the new GCL along with the time lines.

Figure 968. Switching to a new configuration that is aligned with the existing configuration



In the example given in *Table 830*, after the sixth iteration of the first GCL, the BTR values of the old and new GCL are equal. At that point the new GCL is processed as a natural extension to the existing GCL.

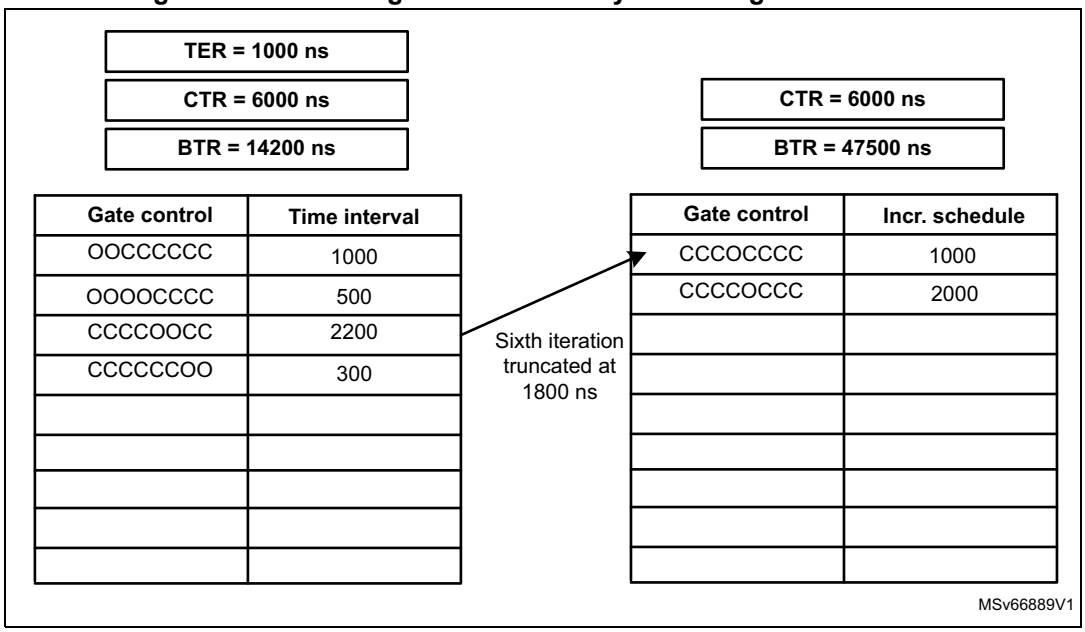
Table 830. GCL and associated registers - BTR and CTR

Current time	Gate control applied	Accumulator value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47900	CCCCCCOO	4000	44200
48200	OOOOOOOO	0	50200
50200	CCCOCCCC	1000	50200
51200	CCCCOCCC	3000	50200
53200	OOOOOOOO	0	56200

New base time unaligned with current schedule

If the new CycleTime differs from the current CycleTime or the new BaseTime is in the future and is not an integer multiple of current CycleTime, then the old and new cycles do not line up. When a new BaseTime is reached (that is when the new configuration is installed and starts to execute), the last old cycle is normally truncated to start the first new cycle. This can be undesirable if it results in a very short last old cycle. It would be better to simply extend the penultimate old cycle by that small amount, rather than starting a very short cycle. The cycle time extension register (TER, related to the current configuration list) enables this extension of the last old cycle to be done in a defined way. If the last complete old cycle ends normally in less than the current cycle time extension (TER in ns) before the new BaseTime, then the last complete cycle before the new BaseTime is reached is extended so that it ends at the new BaseTime.

Figure 969. Switching to the new list by truncating the current list



At the end of the fifth iteration, current time + cycle time extension (TER) < New BTR, so the sixth iteration of the current configuration has started. During the sixth iteration of the current list, when the new BTR value is smaller than the next schedule in the current list, it switches to the new list.

Table 831. Extending to new list by truncating the current list

Current time	Gate control applied	Accumulator value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47500	CCCOCCCC	4000	44200
48500	CCCCOCCC	0	50200
50500	OOOOOOOO	1000	50200

Figure 970 shows an example where the current configuration list is extended instead of starting a new iteration since the extension time of 800 ns is less than the allowed cycle extension time (TER) of 1000 ns.

Figure 970. Switching to new list by extending the current list

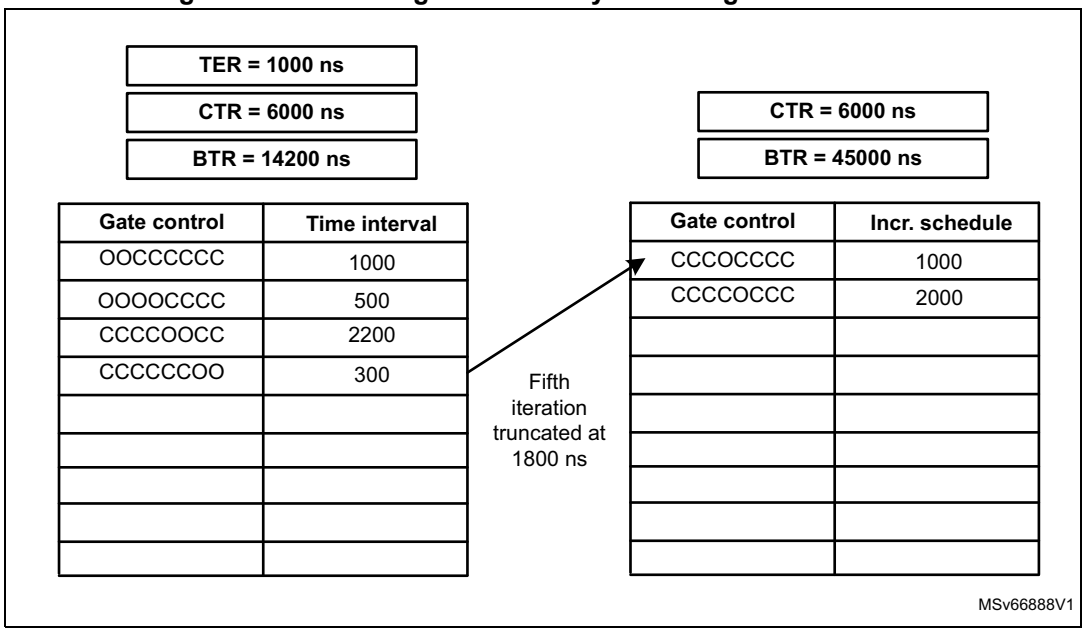


Table 832. Switching to new list by extending the current list

Current time	Gate control applied	Accumulator value	BTR (with updates)
38200	OOCCCCCC	1000	38200
39200	OOOOCCCC	1500	38200
39700	CCCCOOCC	3700	38200
41900	CCCCCCOO	4000	38200
42200	OOOOOOOO	0	44200
45000	CCCOCCCC	1000	45000
46000	CCCCOCCC	3000	45000
48000	OOOOOOOO	0	51000

Impact of transmit scheduling algorithms on EST

When EST is used in isolation, the Gate Control List manages the final open/close state of the queues along with the checks carried out by the transmission selection algorithm in MTL. Since the gate controls operate on a predefined repetitive schedule, it is recommended to use strict priority or credit based shaper (CBS) scheduling algorithms.

Other algorithms such as the weighted round robin (WRR), deficit weighted round robin (DWRR) and weighted fair queuing (WFQ) implement the masking of the queues based on the current winning queue. The algorithm is applied only to the group of queues that open simultaneously. To ensure the queues whose gates are open get priority, these algorithms are modified to treat gate open queues and gate closed queues as separate groups, giving priority to gate open queues.

For example, consider four queues (Q3, Q2, Q1, Q0) with respective weights 4:3:2:1. Q3 and Q2 are in open state in one slot, while Q1 and Q0 are in open state in another slot. In this case, the scheduler works as follows:

1. In the first slot, Q3 and Q2 are serviced in the ratio of 4:3 for the duration during which the slot is open.
2. In the second slot, Q1 and Q0 are serviced in the ratio of 2:1.
3. Fresh arbitration is started every time a slot is opened.

In other words, the traffic does not get distributed in the intended ratio of 4:3:2:1; but as two groups with different ratios, and only for the duration of the slot when the gates are open continuously.

EST programming guidelines

For details about programming guidelines for EST, see *Section 77.9.20: Programming guidelines for EST on page 4462*.

Frame preemption (FPE)

Frame preemption overview

One of the important requirements of IEEE 802.1Qbv-2015 standard is to ensure there are no interfering frames during the scheduled windows that are reserved for high priority traffic. To achieve this goal, a guard band, whose size is equal to the largest possible interfering frame, is required before the window starts. However a larger guard band equates to a less efficient use of network bandwidth.

Frame preemption compliant with IEEE802.1Qbu enables reducing the guard band by breaking the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

Note: Internal loopback is not supported for preempted frames

Description of the frame preemption

Preemption enables one or more higher priority (express) frames to interrupt the transmission of a lower priority (preemptable) frame. The preemptable frame transmission is resumed and completed after the express frame transmission is complete. The following two abstractions of the MAC are used to support frame preemption:

- A preemptable MAC (pMAC), which carries the preemptable traffic
- An express MAC (eMAC), which carries the express traffic

Only the part of the MAC that holds the state during preemption is replicated and represented as pMAC and eMAC. There are two ways for the MAC to put on hold the transmission of the preemptable traffic in the presence of express traffic:

- The MTL scheduler interrupts the preemptable traffic that is currently being transmitted.

  When the preemption capability is enabled, the MAC interrupts the transmission and reception of preemptable frames. A preempted fragment can be followed by zero or more express frames, before the continuation fragments. The preemptable frame can be fragmented any number of times. However, the minimum final and non-final fragment length criterion must be met.

  Interleaving of more than one preemptable packet is not permitted. This implies that if a preemptable packet is fragmented by an express packet, another preemptable packet

cannot be transferred until all the remaining fragments of the first preempted packet are transferred.

- The MTL scheduler prevents starting the transmission of preemptable traffic

  When the preemption capability is disabled, the pMAC entity is disabled and only express traffic is transmitted or received.

Note: Queue0 can be an express queue when FPE is enabled and EST is disabled.

Enabling frame preemption

Enable the frame preemption feature by setting EFPE field of *FPE control and status register (ETH_MACFPECSR)*.

Modifying GCL to support FPE

In the EST-only configuration, the GCL entry has 24 bits of time interval and 8 high order bits representing the gate open/close state requirements as shown in *Table 833*.

Table 833. Gate control list when FPE is disabled

Gate control	Time interval (ns)
OOCCCCCC	T0
OOOOCCCC	T1
CCCCOOCC	T2
CCCCCCOO	T3
I	I
I	I
OCOCOCOO	T126
OOOOCCCC	T127

EST only supports SetGate operations, which implies that the gates are set either to open or close at a given time interval.

However, when both FPE and EST are enabled, the GCL also supports Set-and-Hold-MAC and Set-and-release-MAC operations, in addition to the SetGate operation.

To support hold and release operations, the format of the GCL is slightly changed while configuring and enabling FPE. The first Queue (Q0) is always programmed to carry preemption traffic and therefore it is always open. The bit corresponding to Q0 is renamed as Release/Hold MAC control.

The TX queues whose packets are preemptable are identified by setting the PEC field of *FPE control and status register (ETH_MACFPECSR)*. The GCL bit of the corresponding queue is ignored and considered as always open. As a result, even if the software writes a 0 (C), it is ignored for such queues.

Table 834. Gate control list when FPE is enabled

Gate control (up to 7 bits)	Release/Hold indication	Time interval (ns): 16, 20 or 24 bits
CCCOOOO	0	T0
CCCCOOO	0	T1

Table 834. Gate control list when FPE is enabled (continued)

Gate control (up to 7 bits)	Release/Hold indication	Time interval (ns): 16, 20 or 24 bits
OCCCOOO	1	T2
COCCOOO	1	T3
CCOCOOO	1	T4
CCCOOOO	0	T5
CCCCOOO	0	T6

When the Release/Hold bit transitions from a 0 to 1, a Set-and-Hold-MAC operation is performed. This marks the cease of the preemptable traffic. This is achieved by sending a Hold request to MTL *ha* ns in advance (where *ha* is the time interval mentioned in the Hold advance (HADV) field of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)*.When the Release/Hold bit transitions from a '1' to '0', a Set-and-Release-MAC operation is performed. This marks the resuming of the preemptable traffic. This is achieved by sending a Release request to MTL *ra* ns in advance (where *ra* is the time interval mentioned in the Release Advance (RADV) field of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)*).The preemptable traffic is blocked for the time duration the Release/Hold bit is set to a '1' in the gate control list.

Note: HADV is the advance time to initiate the hold state. Ideally HADV should be programmed to a value of minimum fragment size, which should be smaller than the time interval (minimum fragment size + IPG + Preamble overheads in ns).

Impact of preemption on CBS algorithm

When the CBS algorithm is used, the definition of *Transmit* is as follows:

- TRUE for the duration of frame transmission from the queue;
- FALSE when frame transmission from the queue is complete.

When the CBS algorithm is used along with frame preemption, the value of *Transmit* is TRUE only during the frame transmission by the MAC. If the frame transmission is delayed or interrupted (for instance, a preemptable frame transmission is interrupted to allow the transmission of an express frame from a different queue, or the start of an express frame is delayed because a preemptable frame is being transmitted), the value of *Transmit* is FALSE until the transmission of the frame begins or is resumed.

In addition, the value of *Transmit* is FALSE during the transmission of any overhead resulting from a frame preemption. For example, additional frame overhead (mCRC, Fragment Count) that is added to the preemptable frame.

At any given time, if there are no frames in the queue, the value of *Transmit* is FALSE, and the credit is positive, the credit value is set to zero if there is no preemptable frame from the queue for which transmission is in progress but has been interrupted.

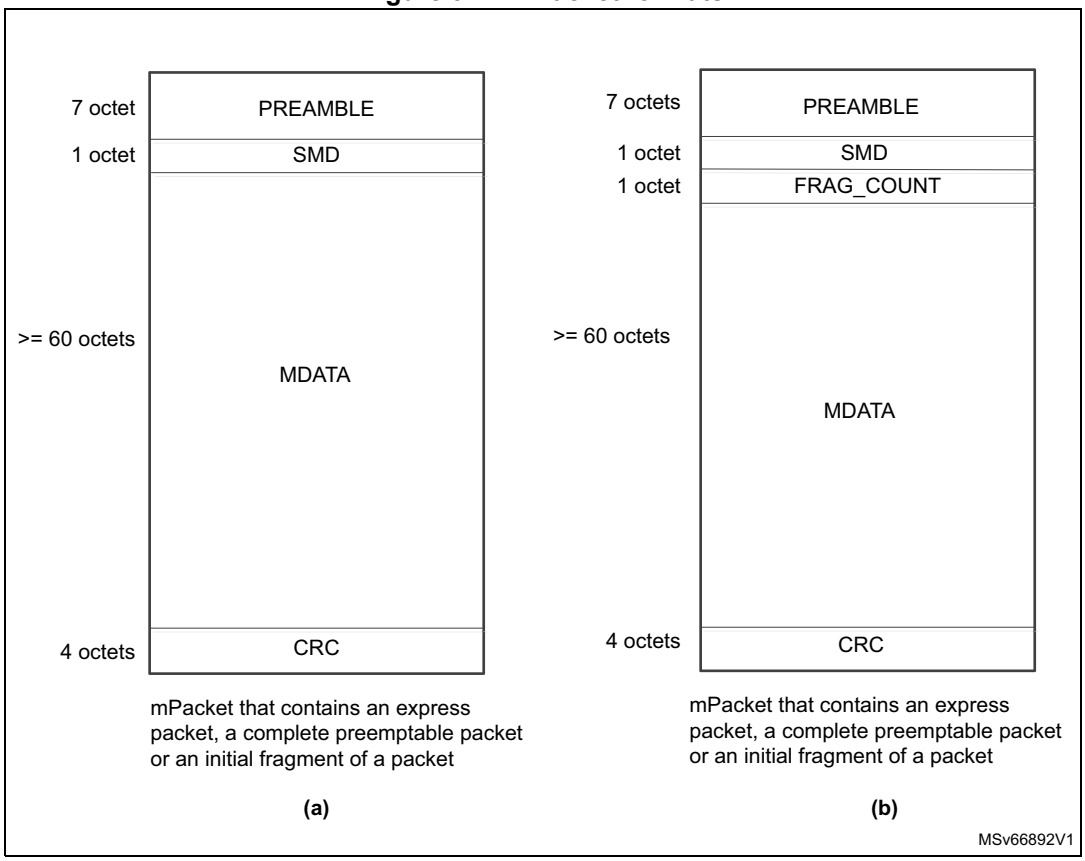
mPacket Format

When the preemption capability is enabled, MAC sends mPackets to the PHY. An mPacket can be:

- An express packet
- A preemptable packet
- An initial fragment of a preemptable packet
- A continuation fragment of a preemptable packet

Figure 971 shows the format of the mPacket. It contains an express packet, a complete preemptable packet or the initial fragment of a preemptable packet. *Figure 971* (b) shows the format of an mPacket containing a continuation fragment of a packet.

Figure 971. mPacket formats



Preamble

The preamble in the mPacket shown in *Figure 971* (a) contains seven octets. The preamble in the mPacket shown in *Figure 971* (b) contains six octets. Each octet contains the value 0x55, transmitted from left to right, 10101010.

Start mPacket Delimiter (SMD)

The value of the SMD indicates whether the mPacket contains an express packet, the start of a preemptable packet (initial fragment or complete packet), or any continuation fragments of a preemptable packet. *Table 835* shows the valid SMD values.

Table 835. Valid SMD Values of mPacket

mPacket type	Notation	Frame count	Value
verify packet	SMD-V	-	0x07
Respond packet	SMD-R	-	0x19
Express packet	SMD-E	-	0xD5
Preemptable packet start	SMD-S0	0	0xE6
	SMD-S1	1	0x4C
	SMD-S2	2	0x7F
	SMD-S3	3	0xB3
Continuation fragment	SMD-C0	0	0x61
	SMD-C1	1	0x52
	SMD-C2	2	0x9E
	SMD-C3	3	0x2A

frag_count

A frag_count is a modulo-4 counter that increments for each continuation fragment of the preemptable packet. The frag_count protects against mPacket reassembly errors by enabling detection of the loss of up to three packet fragments.

The frag_count field is present only in mPackets with SMD-C notation (continuation fragment). The frag_count is zero in the first continuation fragment of each preemptable packet.

Table 836. Valid frag_count values

frag_count	Value
0	0xE6
1	0x4C
2	0x7F
3	0xB3

mData

The contents of the packet from the MAC, starting with the first byte after the SFD to the last byte before the FCS, are sent in the mData fields of one or more mPackets for that frame. The minimum size of the mData field is 60 bytes.

CRC

The CRC field contains a cyclic redundancy check (CRC) and provides an indication on the final mPacket of a frame. In the final mPacket of a frame, the CRC field contains the last four octets of the MAC frame (FCS field).

For other mPackets, the CRC field contains an mCRC value. The mCRC is calculated on the octets of the packet from the first octet of the frame (octet following the SFD of

preemption frames) to the last octet of the packet transmitted in that mPacket, by performing an XOR of the calculated 32-bit CRC value of the fragment with 0x0000FFFF.

Summary of Packet Formats

- Express packet
  Seven bytes of preamble, SMD-E, data and CRC
- Complete Preemptable packet
  Seven bytes of preamble, current preemptable packet SMD, data, CRC
- Initial fragment (non-final) of preemptable packet
  Seven bytes of preamble, current preemptable packet SMD, data, mCRC
- Continuation fragments (non-final) of preemptable packet
  Six bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, mCRC
- Final fragment of preemptable packet
  Six bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, CRC

Note: When frame preemption is enabled, the MAC receiver communicates the exceptions related to received preempted fragments (such as incorrect fragment sequencing or missed fragment) to the MTL layer by reporting it as CRC error. The software should not set the DCRCC (Disable CRC Checking for Received Packets) bit of Extended operating mode configuration register (ETH_MACECR) to 1, otherwise the MTL layer might forward the unintended preempted fragments to the application.

Table 837. Current and Previous SMD Values

Previous preemptable packet SMD	Current preemptable packet SMD
SMD-S0	SMD-S1
SMD-S1	SMD-S2
SMD-S2	SMD-S3
SMD-S3	SMD-S0

Table 838. Current and previous SMD values

Previous preemptable fragment SMD	Previous preemptable fragment FC	Current preemptable continuation fragment SMD	Current preemptable continuation fragment FC
SMD-S0	NA	SMD-C0	FC0
SMD-S1	NA	SMD-C1	FC0
SMD-S2	NA	SMD-C2	FC0
SMD-S3	NA	SMD-C3	FC0
SMD-C0	FC0	SMD-C0	FC1
SMD-C0	FC1	SMD-C0	FC2
SMD-C0	FC2	SMD-C0	FC3
SMD-C0	FC3	SMD-C0	FC0

Table 838. Current and previous SMD values (continued)

Previous preemptable fragment SMD	Previous preemptable fragment FC	Current preemptable continuation fragment SMD	Current preemptable continuation fragment FC
SMD-C1	FC0	SMD-C1	FC1
SMD-C1	FC1	SMD-C1	FC2
SMD-C1	FC2	SMD-C1	FC3
SMD-C1	FC3	SMD-C1	FC0
SMD-C2	FC0	SMD-C2	FC1
SMD-C2	FC1	SMD-C2	FC2
SMD-C2	FC2	SMD-C2	FC3
SMD-C2	FC3	SMD-C2	FC0
SMD-C3	FC0	SMD-C3	FC1
SMD-C3	FC1	SMD-C3	FC2
SMD-C3	FC2	SMD-C3	FC3
SMD-C3	FC3	SMD-C3	FC0

Transmit preemption

MTL Tx preemption

To support preemption, the MAC should have more than one TX queue with at least one queue designated as express queue, plus one queue designated as preemption queue. When FPE is enabled (EFPE set to 1 in *FPE control and status register (ETH_MACFPECSR)*), the MTL preempts a preemptable frame. When a Hold request is asserted (EST configured and enabled), express frames are available for transmission, that is frames are present in the MTL FIFO and qualified for arbitration, after ensuring that the minimum mPacket mData field size is met. Therefore, preemption occurs only if at least 60 bytes of the preemptable frame have been transmitted and at least 64 bytes (including the frame CRC) remain to be transmitted.

The earliest starting position of preemption is controlled by the AFSZ field of *FPE control and status register (ETH_MACFPECSR)*. Preemption does not occur until at least 64 × (1 + AFSZ) - 4 bytes of the preemptable frame have been sent.

When preemption occurs, all the preemptable queues are blocked and only the express queues are allowed to arbitrate (if more than one express queue has traffic) and transmit.

The continuation fragment of the preempted frame is the first frame to be transmitted after a Release request is asserted (EST configured and enabled) and all the express traffic transmission completes.

Note: All the PTP packets should be transmitted as express packets.

MTL should wait for the previous fragment status to be received before resuming the continuation fragments of a preempted frame.

MAC Tx preemption

The MAC supports the preemption by implementing the functionality required to generate the mPackets as described in *Section : mPacket Format*.

When the preemption is enabled, the MAC replaces the SFD of a preemption packet with an SMD-S value. A 2-bit rolling frame count is encoded in the SMD-S value.

The SMD-E value is the same as the SFD value, so the SFD of an express packet does not need to be replaced.

Receive preemption

MAC receive preemption

When FPE is enabled, the MAC receiver passes the incoming packets and differentiates between express packets and preemptable packets. An SMD containing an SMD-E indicates express packet while and SMD containing an SMD-S indicates the first mPacket of a preemptable packet.

If an mPacket containing an SMD-S is received when the reception of the previous preempted packet has not be completed, the MAC sets a CRC error status for the previously received partial packet.

When an SMD-S is detected, the MAC records the frame count indicated by the SMD, and begins sending data on the MRI interface.

The MAC checks the last four bytes of the mPacket. If they do not match CRC, it indicates the end of the packet with or without a CRC error as per the CRC check result. If the last four octets of the mPacket match, it indicates that the packet was preempted.

An SMD containing an SMD-C indicates an mPacket that continues the data for a preempted packet. Upon receiving an SMD value of SMD-C, the MAC checks the following:

- A preempted packet is in progress.
- The frame count indicated by the SMD matches the frame count of the packet in progress.
- The frag_count value indicates the next fragment count.

If any of these checks fails, the mPacket is discarded and the MAC sets a CRC error status for the partially received packet.

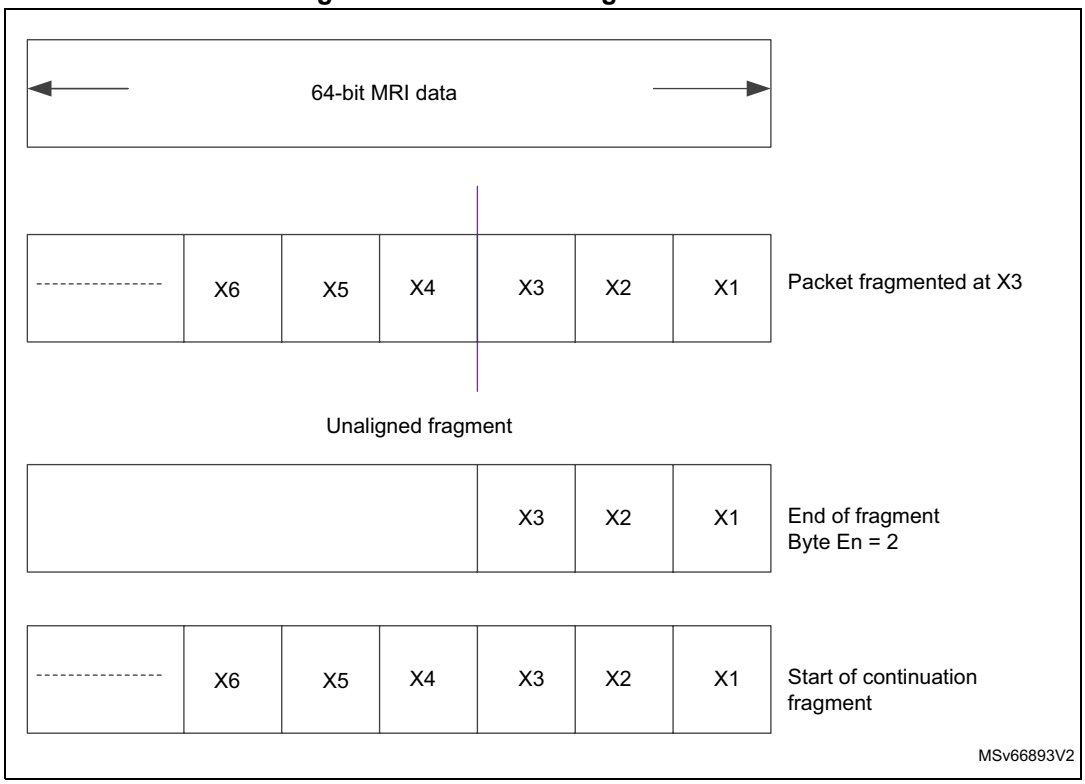
If all the checks pass, the next fragment count is incremented modulo 4.

When a packet is preempted, the MAC saves the state of the partially received packet (such as filter check status, timestamp, length fields) and can process any received express packets before the continuation fragment is received.

Data alignment

When a received frame cannot be fragmented on any byte boundary, the MAC retains the unaligned bytes of data in the previous fragment and resends them with the next fragment as shown in *Figure 972*.

Figure 972. MAC data alignment feature



The MTL can choose to ignore the partial data (based on byte enable value) that comes with end of fragment and use instead the data-width-aligned data received in the first beat of the next continuation fragment.

MTL receive preemption

The MTL Rx must have at least two Rx queues to support FPE function since the preemptable packets and the express packets must be routed to separate Rx queues. The destination Rx queue of a received packet is controlled by ETH_MACRXQCiR registers. Program these registers so that preemptable traffic and express traffic are not routed to the same Rx queue. As the queue mapping for tagged packets is based on VLAN user-priority field, this implies that priority of preemptable and express packets are mutually exclusive. In other words, packets of a certain priority (traffic class) are either express or preemptable but cannot be both.

For the preemptable frames, in addition to the PSRQ (Priority Selected in Rx queue) based queue routing, a programmable frame preemption residue queue (FPRQ) is supported to route all the other received preemption packets (untagged, SA/ DA or VLAN filter fails forwarded due to RA being set or VTFE being reset).

Table 839. Preemption control values on MRI interface for various frame types

Preemption packet type	Queue routing
AV tagged packets passing the SA/DA and VLAN filters	PSRQ(1)
Generic tagged packets passing the SA/DA and VLAN filters	PSRQ(2)
Tagged packets failing the SA/DA and VLAN filters without clearing RA to 0 or setting VTFE to 1	Dropped
All other packets OR when RA is set to 1	FPRQ

1. Only if the queue is enabled for AV feature.
2. Only if the queue is enabled for Generic feature.

The express frames use Queue 0 as a default queue. As a result, Queue 0 must not be used for receiving preemption frames in case of filter failures.

When a fragment is received, it is pushed into its destination Rx queue. When this packet is preempted and is followed by an express packet, it is pushed/stored in a different Rx queue. The MTL saves the context of the preempted frame and processes one or more express frames. When the continuation fragment arrives, it restores the saved context and pushes the remainder of the fragmented frame into its Rx queue.

MTL receive arbitration

On the DMA interface, data is fetched from the MTL Rx FIFO based on the arbitration selected in *Operating mode register (ETH_MTLOMR)*. Frame based arbitration can be used only when all the MTL preemption queues operate in Store and Forward mode. Otherwise, there is a loss of bandwidth on the ARI interface since all the fragments of a preemptable packet are not available. Therefore, express packets received between the fragments are blocked until all the fragments are received and transferred; this defeats the purpose of express traffic.

When operating either in Threshold (cut through) or Store and Forward operating mode, PBL-based arbitration is recommended over frame-based arbitration. If PBL-based arbitration is used, the watermark check is always performed as well as the arbitration/transfer of data in terms of chunks of PBL size of data, which is almost similar to the concepts of fragments. Therefore the express queue packets is also blocked for less time since the DMA interface transfers the data without loss of efficiency.

Received preemption frames support in offload engines

When FPE is enabled, all the packets processed by the offload engines (ARP, PTO) and PMT, PTP, PAUSE, and PFC packets must be received as express/normal packets. These offload engine logic and functions do not recognize preemption frames.

Verify and respond mPackets

When FPE function is present, the MAC can receive and detect the Verify and Respond mPackets, even when FPE is not enabled by software. When the MAC detects valid verify/respond mPackets, it notifies the software by setting the RVER and RRSP fields of *FPE control and status register (ETH_MACFPECSR)*, respectively. Optionally an interrupt can be generated. As such packets have empty (all-zero) data payload, they are dropped inside the MAC and not forwarded to the MRI.

The software can set the SVER and SRSP fields of *FPE control and status register (ETH_MACFPECSR)* to request the MAC to transmit Verify and Respond mPackets respectively. Upon successful transmission of these frames, the MAC clears the SVER/SRSP bits and sets the TVER and TRSP bitfields in *FPE control and status register (ETH_MACFPECSR)*. Optionally an interrupt can be generated when these events occur.

Note: 1. For the preemption frames received with CRC error indication, ignore the following status fields: Packet length, Dribble error indication, Runt frame indication.

2. When FPE is configured, the forward undersized good packets (FUP) bit of ETH_MTLOMR register should not be set.

3. When FPE is configured, do not disable CRC checking on receive. This is required because, FPE error fragments are identified internally using CRC checks. For more details, refer to the DCRCC bitfield in Extended operating mode configuration register (ETH_MACECR).

4. Express and preemption traffic must be routed to different DMAs.

Frame preemption and MMC counter and interrupt registers

The following MMC counters and associated Interrupt registers are present in the MAC.

Table 840. MMC Counters and Associated Interrupt Registers

Frame assembly error counter	Description	Associated MMC counter
Frame assembly error counter	32-bit counter providing the number of MAC frames with reassembly errors on the receiver due to a mismatch in the fragment count value.	*MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)*
Frame SMD error counter	32-bit counter providing the number of received MAC frames rejected because they arrived with an SMD-C when there was no preceding preempted frame	*MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)*
Frame assembly OK counter	32-bit counter providing the number of MAC frames that were successfully reassembled	*MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)*
MAC Rx fragment counter	32-bit counter providing the number of additional mPackets received due to preemption	*MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)*
MAC Tx fragment counter	32-bit counter providing the number of additional mPackets transmitted due to preemption	*MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)*
Hold request counter	32-bit counter containing the number of hold requests sent to the MAC	*MMC Tx hold request counter register (ETH_MMC_TX_HRCR)*

The following additional registers are associated to MMC interrupts for the MMC error counters:

- *MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)*
- *MMC FPE Tx interrupt mask register (ETH_MMC_FPE_TX_IMR)*
- *MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)*
- *MMC FPE Rx interrupt mask register (ETH_MMC_FPE_RX_IMR)*

Time-based scheduling (TBS)

The time-based scheduling feature is suitable for traffic whose periodicity and rate are predictable. To improve the quality-of-service of such traffic:

- The transmit DMA fetches the packet from the host memory for transmission at designated time. This helps the software to set up the Transmit descriptors in advance, even before the packet is ready and available. It reduces the overhead on the software and avoids constant monitoring of the time and preparing descriptors just in time when the packet is targeted to be transmitted.
- The MAC transmits the packets only at the designated/predetermined time even if the packets are fetched in advance. This helps maintaining a constant transmission rate that can be consumed by the receiver, therefore avoiding congestion and excessive buffering in the network.

Description of time-based scheduling

The time-based scheduling feature supports fetching and launching of an Ethernet packet at (or after) a predetermined time. The time-based scheduling is supported only in the following modes and configurations:

- Full duplex mode
- Link speed of 100 Mbps or higher

Note: Do not enable time-based scheduling for channels on which the TSO feature is enabled.

The ESTM mode bit setting impacts the time-based scheduling feature only when the DUT processes the GCL list.

The slot number is meaningful only after the GCL is active. So, start the GCL list before enabling the time-based scheduling (TBS) and enhancements to scheduled traffic (EST) features. If the GCL is not started, all gates are deemed open. In addition, if the packets do not make progress, the Tx queue creates back pressure.

Definitions

The following are the definitions specific to the time-based scheduling feature:

- Launch time: time beyond which the MTL can schedule the packet for transmission.
- GCL slot number (GSN)
- Launch expiry time: time beyond which the packet is dropped by the MTL.
- Fetch time: time beyond which the Tx DMA can schedule a packet fetch from the host memory.

Launch time

The launch time is specified in the enhanced normal descriptor in the DMA configuration. For more details, refer to *Section 77.10.5: Enhanced descriptor for time-based scheduling*.

When the launch time is specified, it is valid only for the specific frame that is scheduled for transmission. The launch time is reset after the transmission of one frame.

The following are the two launch time formats:

- **Normal/absolute format**

  In this format, the launch time is an absolute time value at which the packet is launched for transmission. The launch time is interpreted in the normal/absolute format if the ESTM bit of the *TBS control register (ETH_MTLTBSCR)* is cleared.

The launch time is a 32-bit value where the most-significant eight bits represent the time in seconds and the remaining 24 bits the time in 256 ns. The launch time is compared with the IEEE 1588-based system/PTP time (bits[39:8]) and rolls over after 256 seconds.

The maximum value of the lower 32 bits of system time is 999,999,999 in decimal (0x3B9A C9FF), and it wraps to 0 when reaching this value (representing a full second). Therefore, the maximum value of the lower 24 bits of the launch time (after multiplying by 256) must be 0x3B9AC9.

As the maximum value of the launch time is 256 seconds:

– The launch time is greater than the current system time when its value is between System Time[39:8] and System Time[39:8] + 128 s.

– The launch time is less than the current system time when its value is between System Time[39:8] + 128 s and System Time[39:8] + 256 s, because this is a modulo 256 computation.

- **EST/offset format**

  In this format, the launch time is an offset value relative to the time indicated by the base time register (BTR) of the GCL list provided in the GCL slot number (GSN). The value in the BTR is always updated to the start time of the current loop of the GCL.

  For each packet, the GSN and the launch time values are specified in the Enhanced Transmit descriptor in the DMA configurations, or as control word in the MTL configurations.

  The launch time offset is a 32-bit value where the upper eight bits represent the time in seconds and the remaining 24 bits the time in 256 ns that is added to the BTR value corresponding to the GCL slot number. The value of the launch time offset must be smaller than the value of the cycle time (it is specified in the CTR register that is implemented when EST is enabled).

  If the CTR is greater than or equal to 1 s, the maximum value of the lower 24 bits of launch time offset should be 999 999 999 in decimal (0x3B9A C9FF).

  In this format, the launch time is a 64-bit value, which is interpreted as follows:

$$\text{Launch time} = \text{Launch GSN BTR[63:0]} + (\text{Launch Time Offset[31:0]} \ll 8)$$

  It is compared with the System Time[63:0].

  GSN BTR is the base time value at which the launch GSN loop started execution.

GCL slot number (GSN)

The GCL slot number (GSN) is a modulo 16 count of the GCL loop count. The count is incremented for every new GCL loop. The installation of a new GCL list does not impact the count. The current GCL slot number can be obtained by reading the CGSN field of the *EST Status register (ETH_MTLESTSR)*.

The maximum value of GSN is 15. Therefore GSN values between the current GSN (CGSN field of *EST Status register (ETH_MTLESTSR)*) and CGSN + 8 represent the current or future slots; all the other GSN values are interpreted as elapsed slots or past slots. The GSN value must consequently be between CGSN and CGSN + 8, for a correct interpretation of time.

Launch expiry time

- **Normal/absolute mode**

  In Normal mode, when the LEOV (launch expiry offset valid bit of *TBS control register (ETH_MTLTBSCR)*) is set, the Launch Expiry Offset (LEOS field of *TBS control register (ETH_MTLTBSCR)*) determines the maximum amount of time during which a frame is eligible for launch, starting from the time at which the frame becomes eligible for launch.

  The Launch Expiry Offset is a 24-bit value defined in 256 ns units, with a maximum value of 999,999,999 ns (0x3B9A C9FF).

  $$\text{Launch expiry time} = (\text{Launch Time}[39:8] + \text{LEOS}[32:8]) \times 256\ \text{ns}$$

  The packet with a specific launch time is considered eligible for transmission when the launch time is less than the system time and (if LEOV is set) the system time is less than the launch expiry time.

  When the system time is greater than the launch expiry time, the frame is categorized as expired and is dropped from the MTL FIFO.

Note: For correct interpretation and meaningful operation, the fetch, launch, and launch expiry times should never be set to a value higher than the current system time + 128 s, since such a value is interpreted as time that has already elapsed.

In Full duplex mode, the frames dropped from the MTL FIFO have the Error Summary (bit 15) and Excessive Deferral (bit 3) of the Tx status set.

- **EST/offset mode**

  In EST mode of operation, when the LEOV field of the *TBS control register (ETH_MTLTBSCR)* is set, the Launch Expiry GSN Offset (LEGOS field of the *TBS control register (ETH_MTLTBSCR)*) and Launch Expiry Offset (LEOS field of *TBS control register (ETH_MTLTBSCR)*) determine the maximum amount of time during which a frame remains eligible for launch, starting from the time at which the frame becomes eligible for launch.

  $$\text{Launch Expiry Offset} = \text{LEGOS} / \text{LEOS}$$

  LEGOS holds the GSN offset (multiples of CTR time) and LEOS holds the maximum value of CTR (subCTR values) in ns.

  The Launch Expiry Offset is a 24-bit value defined in units of 256 ns, with a maximum value which is the smaller of 999,999,999 ns and CTR - 1 ns.

  The Launch Expiry GSN is computed as follows:

  $$\text{Launch Expiry GSN} = \text{Launch GSN} + \text{LEGOS} + \text{CCMA}$$

  Where:

  CCMA is the CTR carry due to modulo addition. This value is 1 if ((Launch Time Offset + LEOS) << 8) is equal to or greater than CTR.

  – When CCMA = 1: Launch Expiry Offset = (Launch Time Offset + LEOS) - CTR.
  – When CCMA = 0, Launch Expiry Offset = (Launch Time Offset + LEOS).

The launch expiry time is computed as follows:

$$\text{Launch expiry time} = \text{Launch Expiry GSN BTR}[63:0] + \text{Launch Expiry Time Offset}$$

When LEOV is not set, the launch expiry time is not checked.

When LEOV is set:

- If the system time is greater than the launch expiry time, the frame is dropped from the MTL FIFO. The frame is considered as expired.
- If the launch time is smaller than the system time and the launch expiry time is greater than the system time, the frame is considered eligible for launch.

Note: The maximum value of LEGOS is 7. This implies that when LEOV is set, the frame has a maximum life time of less than 8 GCL loop iterations after it becomes eligible for launch.

The slot number of the first GCL list executed each time after EST is enabled, is zero.

Fetch time

The fetch time can also be indicated by the software for each packet. This is done by setting the FTOV field of DMA TBS Control Register x (ETH_DMATBSCTRL0R).

If the FTOV filed is not set, the Fetch Time Offset is not valid and the DMA fetches packets without any time constraints.

The fetch time accounts for all possible delays in the DMA fetch operation and ensures that the frame is present in the MTL FIFO before the launch time.

- **Normal/absolute mode**

  In Normal mode, the fetch time is derived/calculated by reducing the time specified in the Fetch Time Offset (FTOS) field of DMA TBS Control Register x (ETH_DMATBSCTRL0R) from the given launch time.

  In Normal mode of operation, the fetch launch time is computed as follows:

  $$\text{Fetch time}[39:8] = \text{Launch Time}[39:8] - \text{FTOS}[31:8]$$

  The fetch time is a 32-bit value. It is compared against System Time[39:8] to determine the eligibility for fetching the frame:

  – The fetch time is defined as greater than the system time if the fetch time is in the range of System Time[39:8] and System Time[39:8] + 128 s. The frame is considered as not-eligible for fetch.

  – The fetch time is defined as smaller than the system time if the fetch time is in the range of System Time[39:8] + 128 s and System Time[39:8] + 256 s. The frame is considered as eligible for fetch.

  This is a modulo 256 computation.

- **EST/offset mode**

  In EST/offset mode, the Fetch GSN Offset (FGOS field of DMA TBS Control Register x (ETH_DMATBSCTRL0R) provides the slot number offset to be deducted from the launch GSN. In this case, the FTOS value should be the smaller of 999,999,999 ns and CTR – 1 ns.

  If Launch Time Offset ≥ FTOS:

  – Fetch Time Offset = ((Launch Time Offset – FTOS) × 256 ns)

  – CBFS (CTR Borrow for Fetch Subtraction) = 0

  If Launch Time Offset < FTOS:

  – Fetch Time Offset = CTR + ((Launch Time Offset – FTOS) × 256 ns)

  – CBFS (CTR Borrow for Fetch Subtraction) = 1

  The Fetch GSN is computed as follows:

  – Fetch GSN = Launch GSN – FGOS – CBFS

  – Fetch Time = Fetch GSN BTR[63:0] + Fetch Time Offset

Note: The frame is marked eligible for DMA fetch when the fetch time is smaller than the system time.

The maximum value of FGOS is 7. This implies that when FTOV is set, the frame can be fetched at a maximum of 8 GCL loop iterations before it becomes eligible for Launch.

- **DMA operation sequence**

  The following is the sequence of operation when FTOV is set:

  a) Fetch the first enhanced normal descriptor (FD is set).

  b) If LTV is set and fetch enabled, compute the fetch time based on the launch time. Wait for the system time to be greater than the fetch time

  c) Read the frame (data) from the host memory and transfer it to the MTL FIFO.

  d) Close the normal descriptor.

  e) Fetch the next normal descriptor (if the previous descriptor was not the last).

  f) Repeat steps **d** to **f** until the last descriptor of the frame (LD is set).

  After the last descriptor of a frame, the next enhanced normal descriptor should be programmed with a new launch time and with LTV bit set. Otherwise, the subsequent frames are processed without any time restrictions.

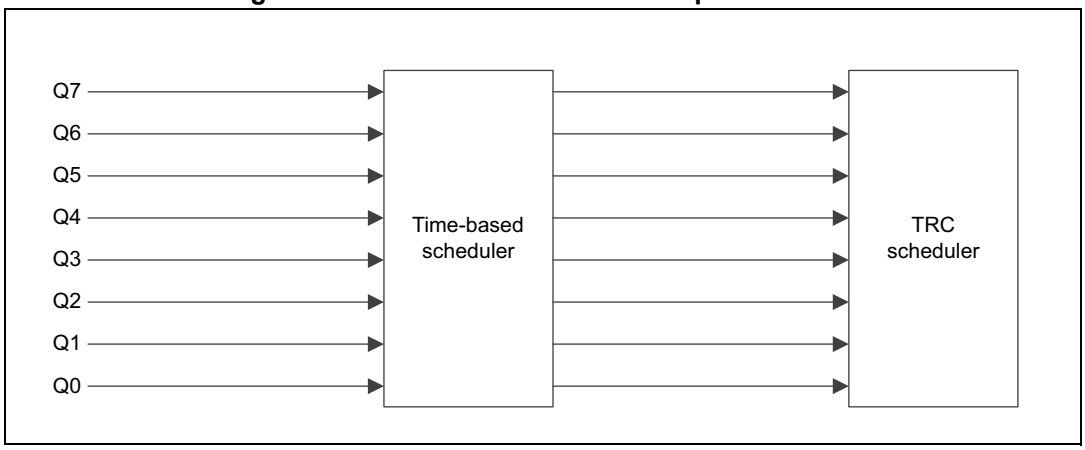
- **Impact on the transmission selection algorithm**

  The time-based scheduler does not directly influence the transmission selection algorithm but has an indirect influence as it determines if a queue is eligible to participate in the scheduling.

  If a frame of a specific queue has a valid launch time, the time-based scheduler ensures that the frame participates in transmit scheduling (TRC scheduler) only after the launch time has elapsed.

  The gating done by the time-based scheduler is only for the scheduling (picking frame for transmission). It might not alter any other characteristics and behavior of the frame. For example, the frame waiting for the launch time to elapse continues to gain credits in CBS, and the CBS credits are not lost as the frame is marked as available but not ready. Strict priority and CBS are recommended transmit scheduling algorithms for the TBS implementation because scheduling for TBS enabled queues is more predictable.

Figure 973. Time-based scheduler implementation



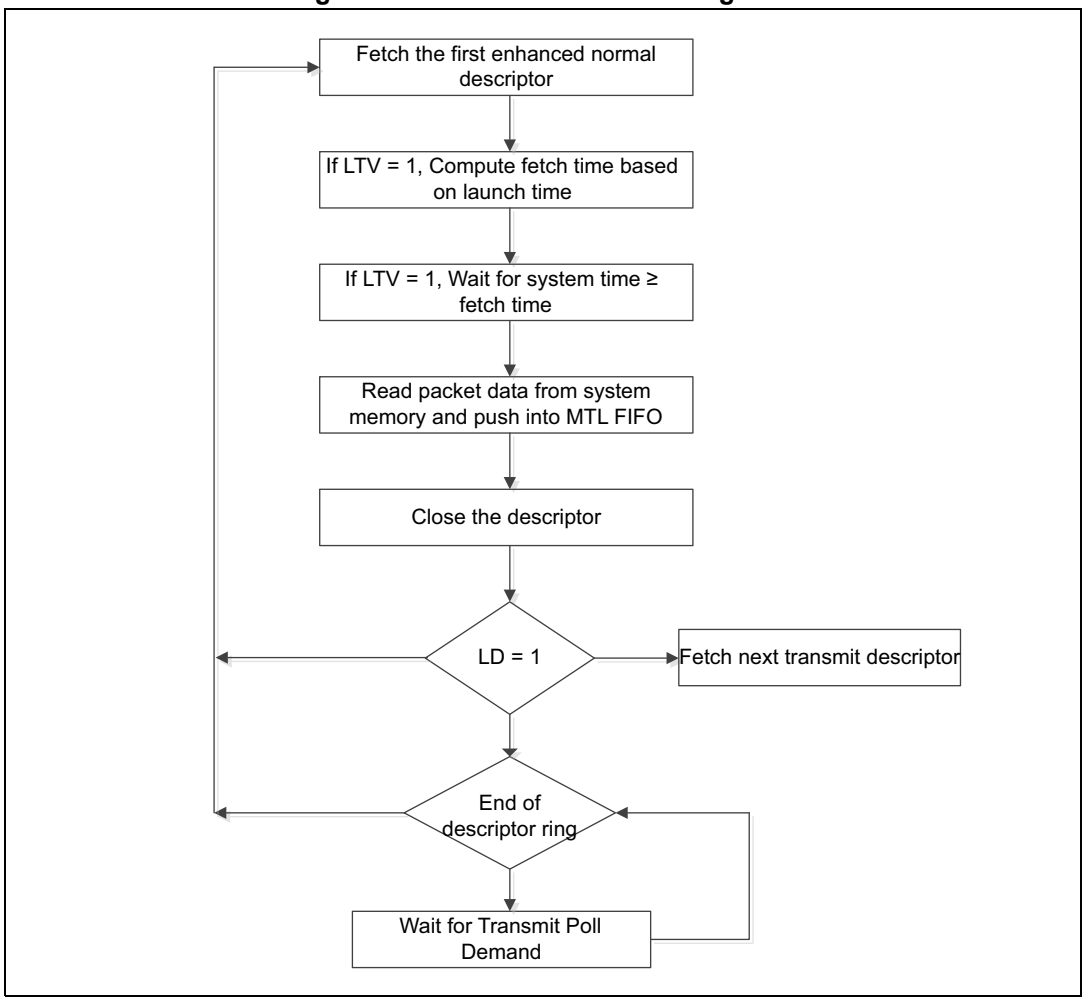
Programming guidelines for time-based scheduling

For details about programming guidelines, see *Section 77.9.21: Programming the launch time in time-based scheduling*.

Time-based scheduling flow

Figure 974 shows the time-based scheduling flow.

Figure 974. Time-based scheduling flow



77.5.7 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO

automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

$$\text{Packet size} < \text{TxQSize} - (\text{PBL} + 5) \times 9$$

where

TxQSize corresponds to the TQS bitfield of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*

PBL corresponds to the TxPBL bitfield of *Channel x transmit control register (ETH_DMACxTXCR)*

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 862: TDES3 normal descriptor (write-back format)*).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 862: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 841 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section : Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.

Table 841. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Not applicable	Yes

Table 841. Transmit checksum offload engine functions for different packet types (continued)

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – No – No – Yes – No – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The Receive Checksum Offload Engine (Rx COE) can be enabled by setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the *VLAN tag Control register (ETH_MACVTCR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 842: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 842. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

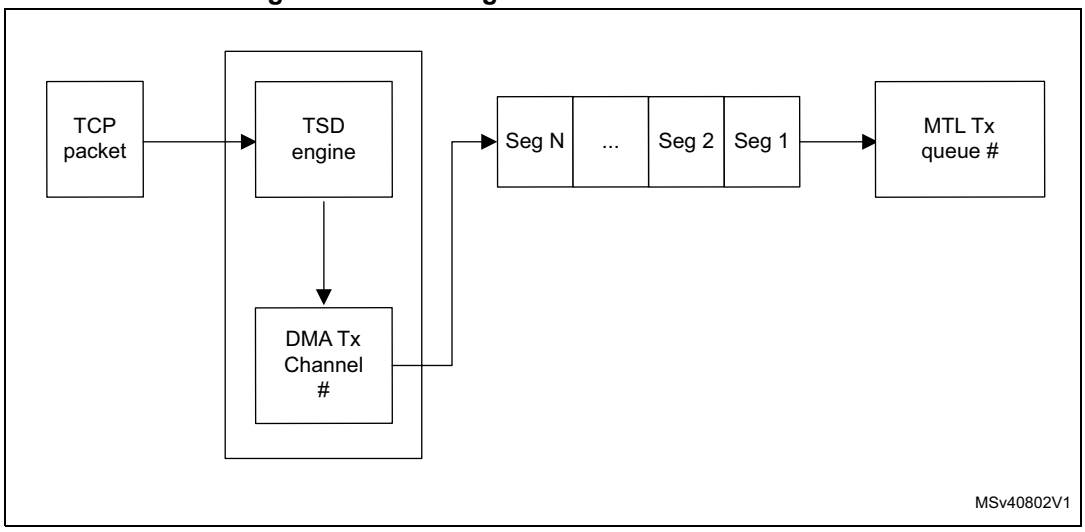
77.5.8 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 975*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACxCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to *Section 77.9.16: Programming guidelines for TSO*.

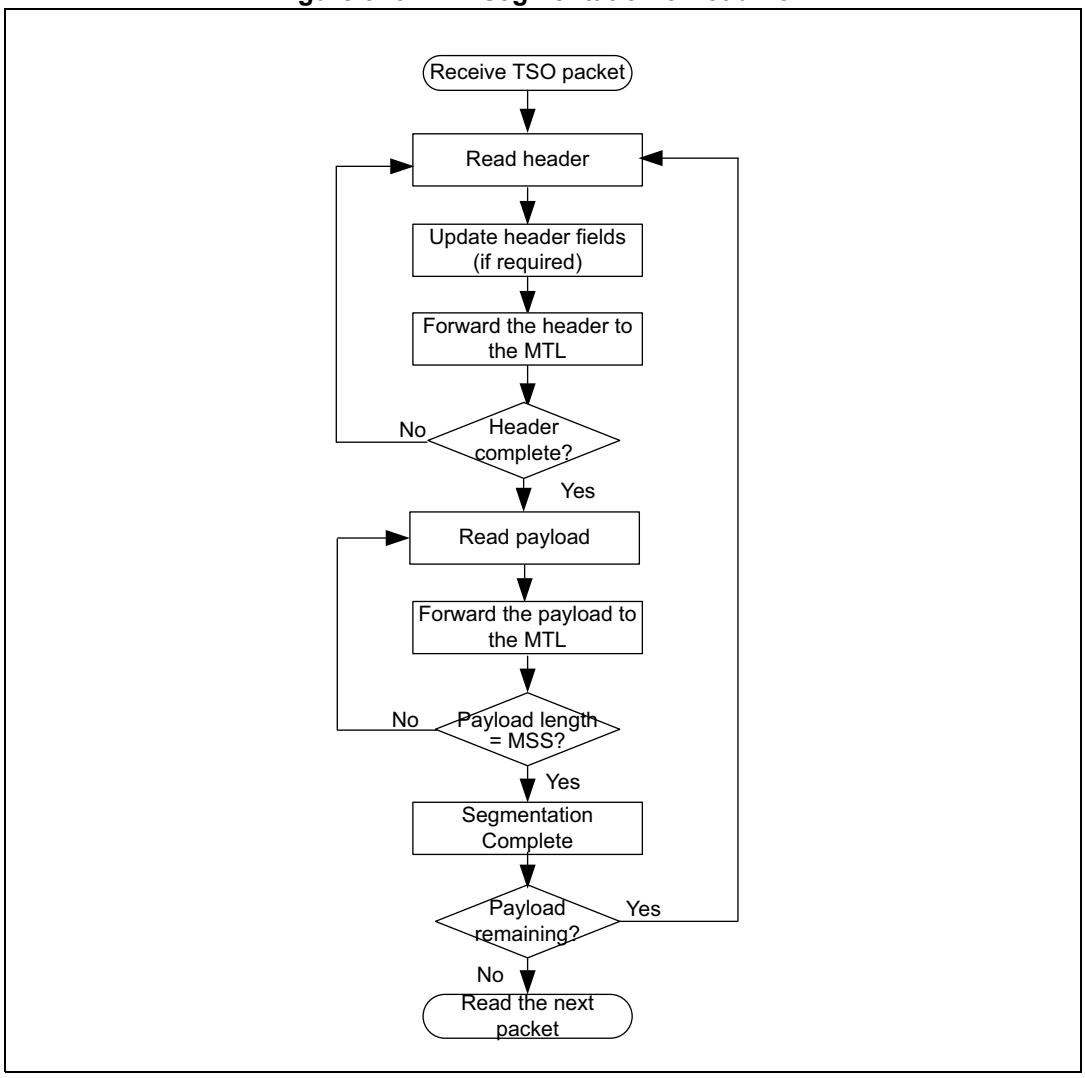
Figure 975. TCP segmentation offload overview



DMA operation with TSO feature

Figure 976 shows the TSO flow.

Figure 976. TCP segmentation offload flow



For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   – If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.

- If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.
- If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 843* describes how the TCP and IP headers are updated.

Table 843. TSO: TCP and IP header fields

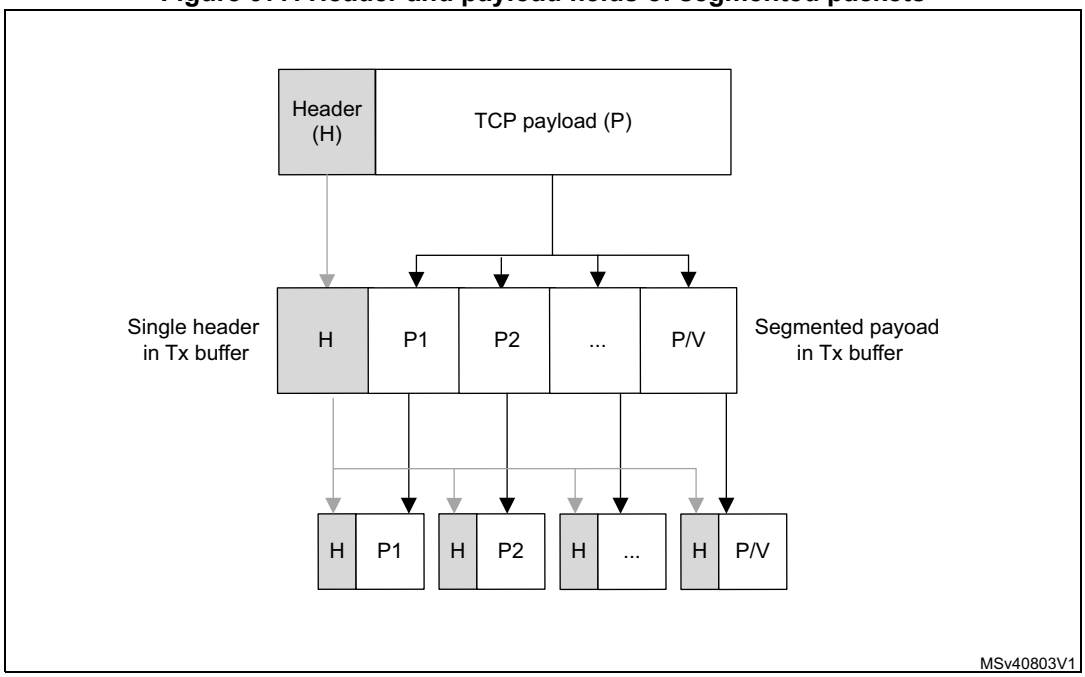
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 843: TSO: TCP and IP header fields*. *Figure 977: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 977. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 77.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel x interrupt enable register (ETH_DMACxIER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel x status register (ETH_DMACxSR)*).

Building the Descriptor and the packet for the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 77.10.3: Transmit descriptor*). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

Note: The TDES3[22:19] bitfield is used as slot number in case of AV configurations. If both TSO and AV are enabled for the same DMA Tx channel (TSE bit is set in Channel x transmit control register (ETH_DMACxTXCR), then do not enable slot checking for that DMA Tx channel (do not set ESC bit in Channel x slot function control status register (ETH_DMACxSFCSR).

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACxCR (see *Channel x control register (ETH_DMACxCR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACxTXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQxOMR register (see *Tx queue x*

operating mode register (ETH_MTLTXQxOMR)). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

77.5.9 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier

ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the *Extended operating mode configuration register (ETH_MACECR)* is set, then the MAC does not check for valid CRC of a ARP request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

77.5.10 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the Receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of two or four (in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between (G)MII Transmit and Receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the *Operating mode configuration register (ETH_MACCR)*. Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII or GMII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on gmii_txd signal.

Preemption is not supported in loopback mode.

77.5.11 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and back-pressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 844* describes the fields of a Pause packet.

Table 844. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 845 describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*
- TFE bit of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 845. Tx MAC flow control

EHFC	TFE	DM	Description
X	0	X	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx performs backpressure when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.
0	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx sends a Pause packet when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.

Transmit flow control

The transmit flow control is enabled when TFE bit is set in *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*.

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding *Tx Queue 0 flow control register (ETH_MACQ0TX-FCR)*.

The flow control is also triggered based on Rx Queue Threshold: the flow control operation of the MAC is enabled when the EHFC bit in the corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in RFA field of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*. This flow control signal is de-asserted when the fill-level of the queue is lower than the threshold configured in the RFD field.

Note: The RFA sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter.

The remote transmitter can continue to send packets until it receives this PAUSE packet. As a results, the RXQ needs space to take in this data even after the flow control is triggered, in order to avoid overflow and loss of packets.

The maximum space required is calculated theoretically. It can be 2 max-sized packets + a little more assuming that the read from the RxFIFO is not occurring during this whole time.

Theoretically, this makes RFA = 4 or more, which implies that RxQ must have a minimum size of 4 Kbytes.

Practically, the system works with RFA = 2 (full size – 2 Kbytes, with very little probability of overflow).

The RFA bitfield in Rx queue x operating mode register (ETH_MTLRXQxOMR) is used for activating the flow control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated.

With an RXQ size of 4 Kbytes, RFA must be set to 2 Kbytes to receive the whole packet.

Receive flow control

In the Receive path, the flow control is functional only in Full-duplex mode. If any Pause packet is received in Half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 846 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of *Rx flow control register (ETH_MACRXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Table 846. Rx MAC flow control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) and the UP bit of *Rx flow control register (ETH_MACRXFCR)* is set.

     If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
   - Type field: this field is checked for 0x8808.
   - Opcode field: this field is checked for 0x0001 (Pause packet).
   - Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

Enabling receive flow control

Set the RFE bit in the *Rx flow control register (ETH_MACRXFCR)* to enable the Pause flow control.

77.5.12 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 77.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the *Packet filtering control register (ETH_MACPFR)*. The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- 52 registers used for collision, error, and good packet or octet counters:
  - *Tx octet count good bad register (ETH_TX_OCTET_COUNT_GOOD_BAD)*
  - *Tx packet count good bad register (ETH_TX_PACKET_COUNT_GOOD_BAD)*
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multicast packets good register (ETH_TX_MULTICAST_PACKETS_GOOD)*
  - *Tx 64 octets packets good bad register (ETH_TX_64OCTETS_PACKETS_GOOD_BAD)*
  - *Tx 65 to 127 octets packets good bad register (ETH_TX_64TO127OCTETS_PACKETS_GOOD_BAD)*
  - *Tx 128 to 255 octets packets good bad register (ETH_TX_128TO255OCTETS_PACKETS_GOOD_BAD)*
  - *Tx 256 to 511 octets packets good bad register (ETH_TX_256TO511OCTETS_PACKETS_GOOD_BAD)*
  - *Tx 512 to 1023 octets packets good bad register (ETH_TX_512TO1023OCTETS_PACKETS_GOOD_BAD)*
  - *Tx 1024 to max octets packets good bad register (ETH_TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)*
  - *Tx unicast packets good bad register (ETH_TX_UNICAST_PACKETS_GOOD_BAD)*
  - *Tx multicast packets good bad register (ETH_TX_MULTICAST_PACKETS_GOOD_BAD)*
  - *Tx broadcast packets good bad register (ETH_TX_BROADCAST_PACKETS_GOOD_BAD)*
  - *Tx underflow error packets register (ETH_TX_UNDERFLOW_ERROR_PACKETS)*
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)*

  - *Tx deferred packets register (ETH_TX_DEFERRED_PACKETS)*
  - *Tx late collision packets register (ETH_TX_LATE_COLLISION_PACKETS)*
  - *Tx excessive collision packets register (ETH_TX_EXCESSIVE_COLLISION_PACKETS)*
  - *Tx carrier error packets register (ETH_TX_CARRIER_ERROR_PACKETS)*
  - *Tx octet count good register (ETH_TX_OCTET_COUNT_GOOD)*
  - *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)*
  - *Tx excessive deferral error register (ETH_TX_EXCESSIVE_DEFERRAL_ERROR)*
  - *Tx pause packets register (ETH_TX_PAUSE_PACKETS)*
  - *Tx VLAN packets good register (ETH_TX_VLAN_PACKETS_GOOD)*
  - *Tx OSsize packets good register (ETH_TX_OSIZE_PACKETS_GOOD)*
  - *Rx packets count good bad register (ETH_RX_PACKETS_COUNT_GOOD_BAD)*
  - *Rx octet count good bad register (ETH_RX_OCTET_COUNT_GOOD_BAD)*
  - *Rx octet count good register (ETH_RX_OCTET_COUNT_GOOD)*
  - *Rx broadcast packets good register (ETH_RX_BROADCAST_PACKETS_GOOD)*
  - *Rx multicast packets good register (ETH_RX_MULTICAST_PACKETS_GOOD)*
  - *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)*
  - *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)*
  - *Rx runt error packets register (ETH_RX_RUNT_ERROR_PACKETS)*
  - *Rx jabber error packets register (ETH_RX_JABBER_ERROR_PACKETS)*
  - *Rx undersize packets good register (ETH_RX_UNDERSIZE_PACKETS_GOOD)*
  - *Rx oversize packets good register (ETH_RX_OVERSIZE_PACKETS_GOOD)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
  - *Rx 65 to 127 octets packets good bad register (ETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD)*
  - *Rx 128 to 255 octets packets good bad register (ETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD)*(
  - *Rx 256 to 511 octets packets good bad register (ETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD)*
  - *Rx 512 to 1023 octets packets good bad register (ETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD)*
  - *Rx 1024 to max octets packets good bad register (ETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
  - *Rx length error packets register (ETH_RX_LENGTH_ERROR_PACKETS)*
  - *Rx out of range type packets register (ETH_RX_OUT_OF_RANGE_PACKETS)*
  - *Rx pause packets register (ETH_RX_PAUSE_PACKETS)*
  - *Rx FIFO overflow packets register (ETH_RX_FIFO_OVERFLOW_PACKETS)*
  - *Rx VLAN packets good bad register (ETH_RX_VLAN_PACKETS_GOOD_BAD)*
  - *Rx watchdog error packets register (ETH_RX_WATCHDOG_ERROR_PACKETS)*
  - *Rx receive error register (ETH_RX_RECEIVE_ERROR)*
  - *Rx control packets good register (ETH_RX_CONTROL_PACKETS_GOOD)*
- Four registers to record LPI mode transition:
  - *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)*
  - *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)*

- *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)*
- *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)*

Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)
  - GMII_RXER input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Table 847*.

Table 847. Size of the maximum receive packet

JE	S2KP	GPSLCE	EDVLP	Untagged frame maximum size in bytes	Single VLAN frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

77.5.13 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

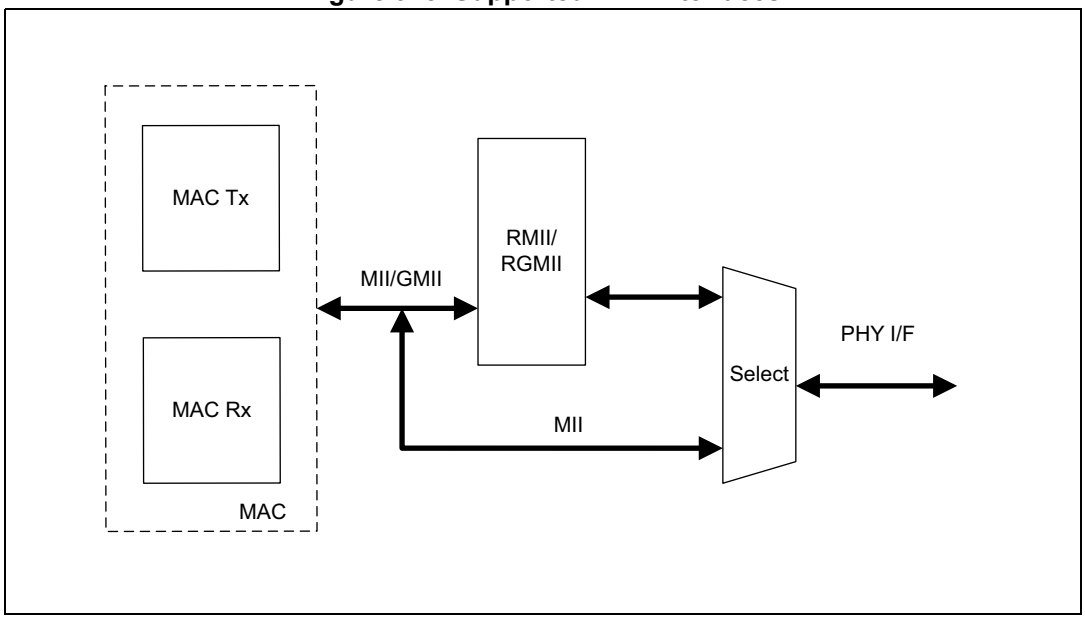
77.5.14 MAC and MMC register descriptions

Refer to *Section 77.11.4: Ethernet MAC and MMC registers*.

77.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII/GMII one. All other interfaces are derived from it as shown in *Figure 978*.

Figure 978. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

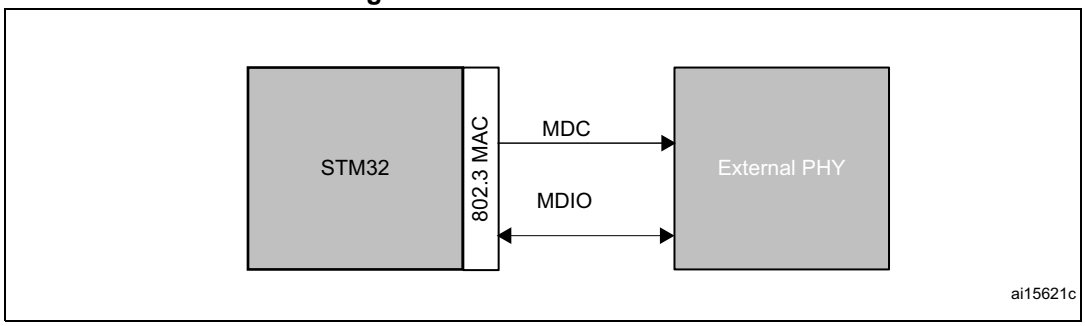
- *Station management agent (SMA)*
- *Media independent interface (MII)*
- *Reduced media independent interface (RMII)*
- *Reduced gigabit media independent interface (RGMII)*

77.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 979*.

Figure 979. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

Table 848. MCD clock selection

Selection	MDC clock
0000	eth_hclk/42
0001	eth_hclk/62
0010	eth_hclk/16
0011	eth_hclk/26
0100	eth_hclk/102
0101	eth_hclk/124
0110	eth_hclk/204
0111	eth_hclk/324
1000	eth_hclk/4
1001	eth_hclk/6
1010	eth_hclk/8
1011	eth_hclk/10
1100	eth_hclk/12
1101	eth_hclk/14
1110	eth_hclk/16
1111	eth_hclk/18

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

Figure 980 shows the structure of a Clause 45 MDIO packet, while *Table 849* provides a detailed description of the packet fields.

Figure 980. MDIO packet structure (Clause 45)

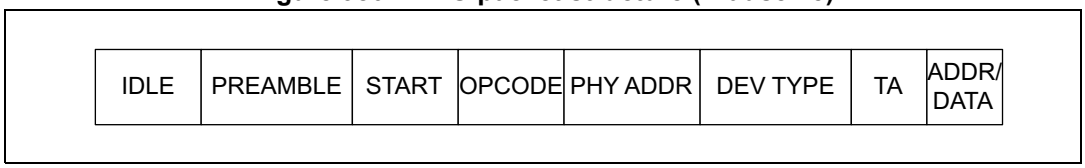


Table 849. MDIO Clause 45 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b00
OPCODE	– 0b00: Address – 0b01: Write – 0b10: Read+ Address – 0b11: Read
PHY ADDR	5-bit address select for one of 32 PHYs
DEV TYPE	5-bit device type
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. – In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits. – In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in *MDIO address register (ETH_MACMDIOAR)* can be programmed to enable Clause 22 or Clause 45 mode of operation. *Figure 981* shows the structure of a Clause 22 MDIO packet, while *Table 850* provides a detailed description of the packet fields.

Figure 981. MDIO packet structure (Clause 22)

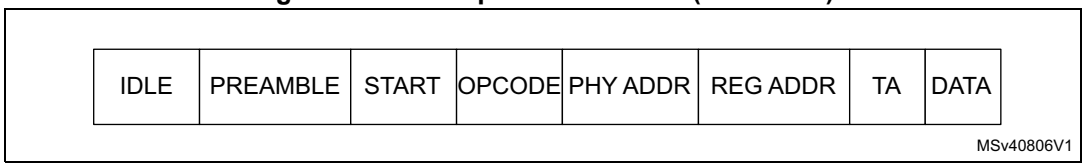


Table 850. MDIO Clause 22 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b01
OPCODE	0b10 for Read and 0b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.

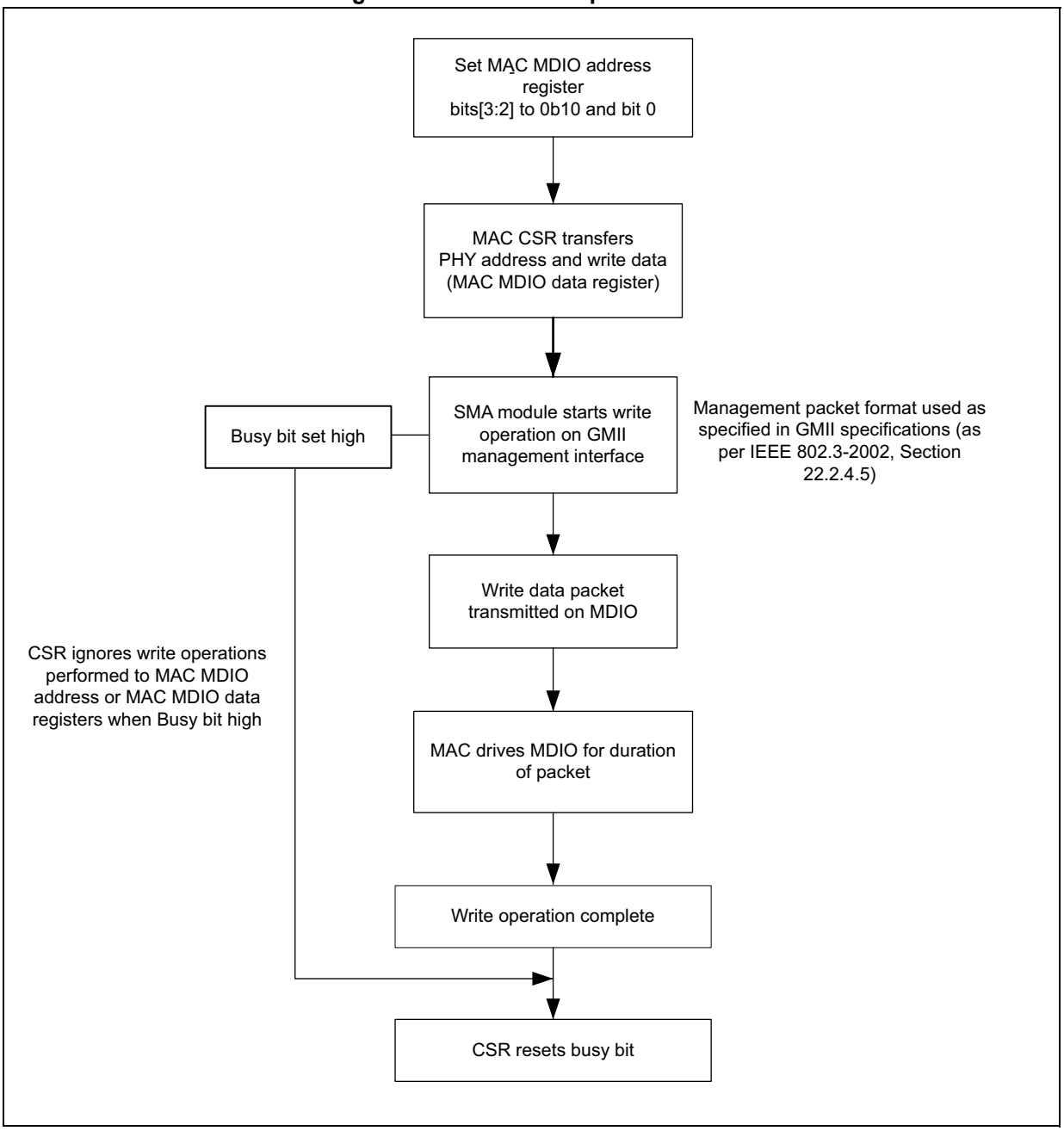
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

GMII/MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 982 illustrates the flow for a write operation from the SMA module to the PHY registers.

Figure 982. SMA write operation flow



When bits[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations per-

formed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register (ETH_MACMDIODR)* during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 983. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

GMII/MII management read operation

When bits[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII/MII management interface using the management packet format specified in the GMII/MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 984. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

MSv40808V1

Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

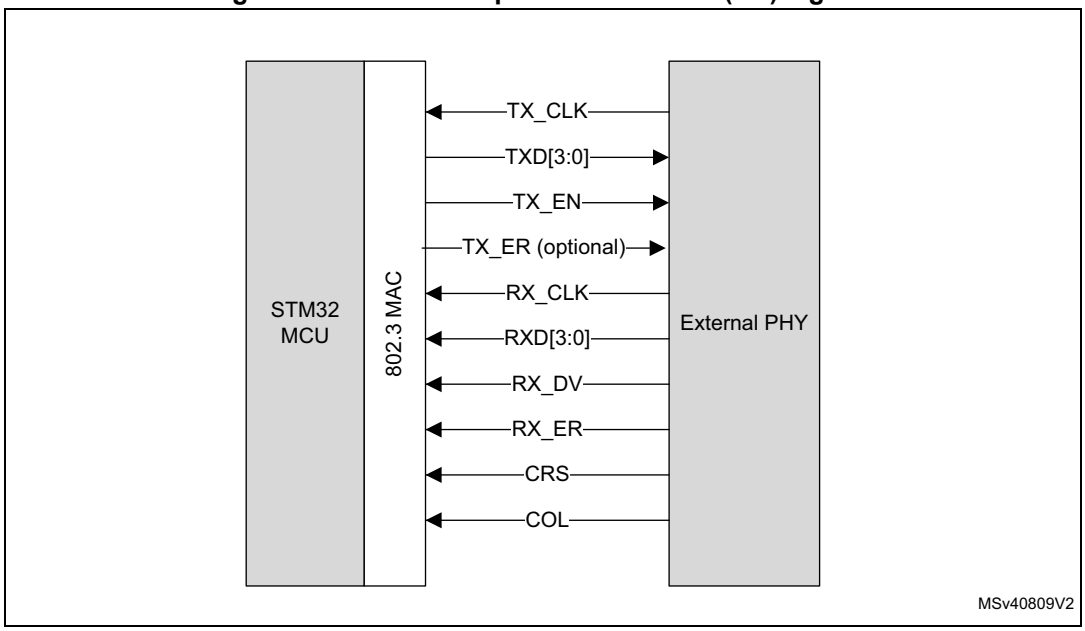
The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.

77.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 985: Media independent interface (MII) signals*.

Figure 985. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is

the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.

- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don't care for the MAC sublayer.

- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don't care for the MAC sublayer.

77.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU (only in 10/100 mode). According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

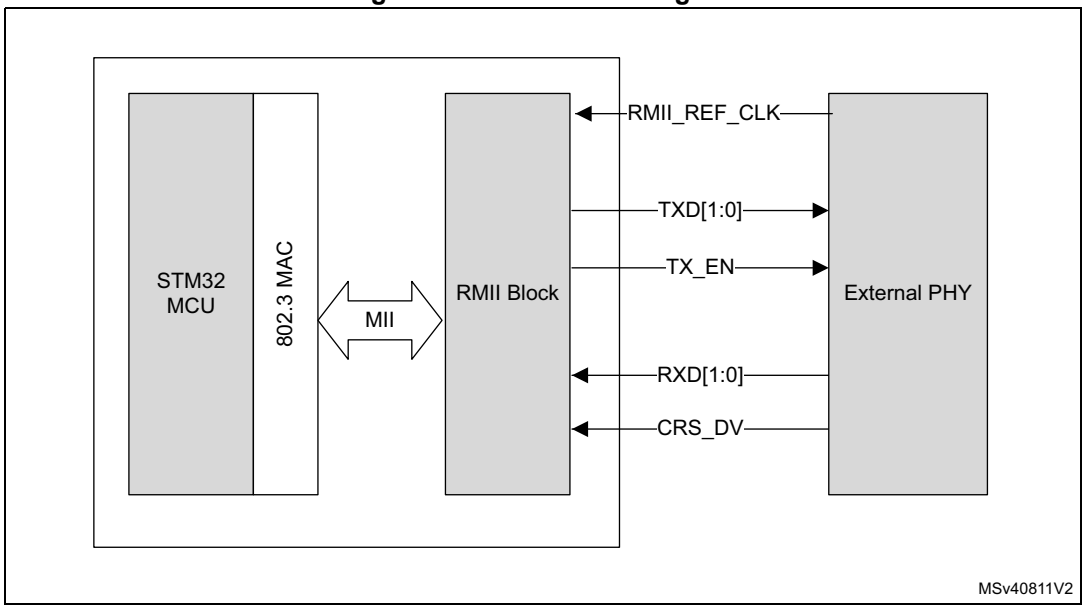
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

RMII block diagram

Figure 986: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

Figure 986. RMII block diagram

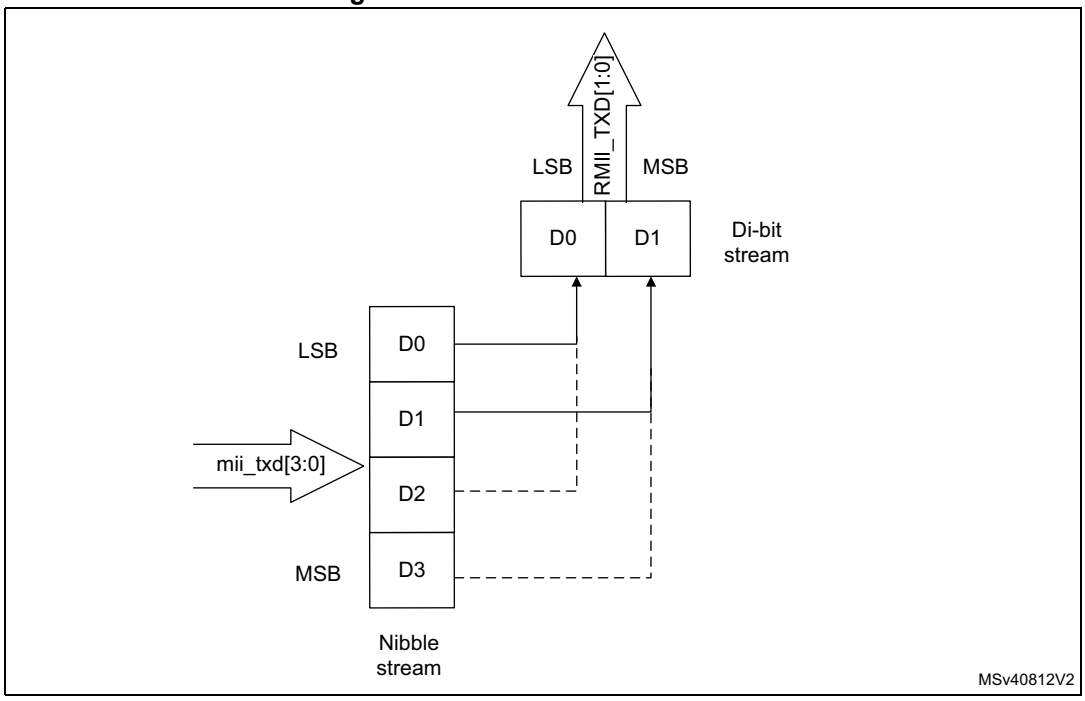


- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 987: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

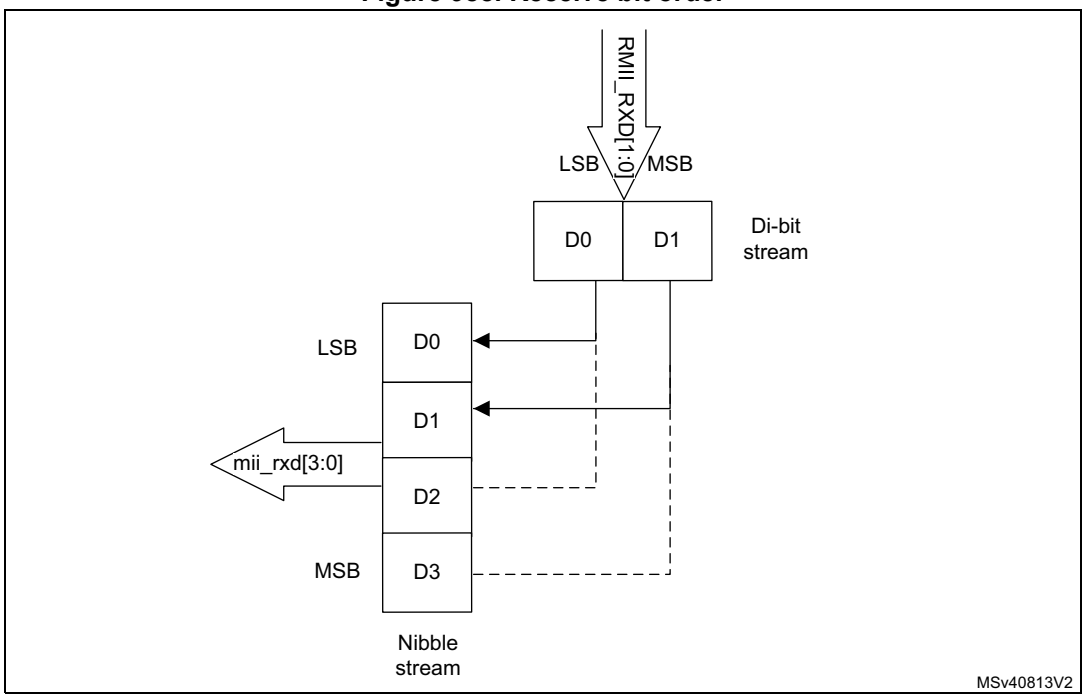
Figure 987. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 988: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 988. Receive bit order



77.6.4 Reduced gigabit media independent interface (RGMII)

The reduced gigabit media independent interface (RGMII) specification reduces the pin count of the interconnection between the MAC and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both edges of the Transmit and Receive clocks. For Gigabit operation, the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5/25 MHz.

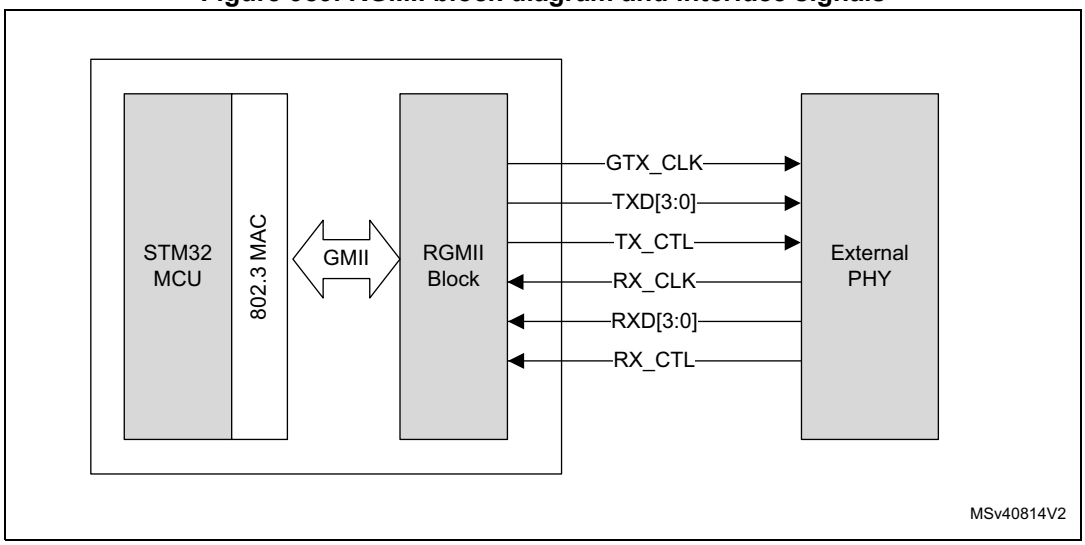
The RGMII module is instantiated between the GMII of the MAC and the PHY to translate the control and data signals between the GMII and RGMII protocols. The RGMII block has the following characteristics:

- Support of 10, 100 and 1000 Mbps operation rates.
- No extra clock required since both edges of the incoming clocks are used.
- Extraction of the in-band (link speed, link status, and duplex mode) status signals from the PHY and provides them to the MAC for link detection.

RGMII block diagram

Figure 989: RGMII block diagram and interface signals shows the position of the RGMII block relative to the MAC and RGMII PHY. The RGMII block is placed between the GMII and the PHY to translate the GMII signals to RGMII signals.

Figure 989. RGMII block diagram and interface signals



- **Transmit path**: this block translates all GMII transmit signals to RGMII signals. The block registers all GMII input signals at the rising edge of eth_gmii_gtx_clk and generates all RGMII transmit signals at the rising and falling edges of eth_gmii_gtx_clk.
- **Receive path:** this block translates all RGMII receive signals to GMII receive signals. The block registers all RGMII input signals at the rising and falling edges of eth_rx_clk and generates all GMII receive signals at the rising edge of eth_rx_clk.
- GTX_CLK: RGMII clock for Tx data transfers, 125 MHz, 25 MHz or 2.5 MHz
- TXD[3:0]: transmit data
- TX_CTL: multiplexing of TX_EN on the rising edge of GTX_CLK, and a logical XOR between TX_EN and TX_ER values on falling edge of GTX_CLK
- RX_CLK: RGMII clock for Rx data transfers
- RXD[3:0]: receive data
- RX_CTL: after demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).

Signal conversion

The RGMII block performs data conversions in both directions.

Transmit data conversion

As defined in the RGMII specification, multiplexing of data and control in Gigabit mode is accomplished by using both edges of the Transmit clock. The RGMII block accepts the 8-bit GMII transmit data from the MAC GMII (gmii_txd[7:0]) on the rising edge of eth_gmii_gtx_-clk. For Gigabit operation, the RGMII block then converts gmii_txd[7:0] to two 4-bit nibbles and transmits the data on the RGMII transmit data port (rgmii_txd_o[3:0]) as follows:

- At the rising edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[3:0]
- At the falling edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[7:4]

For 10/100 mode, the Transmit data to the PHY is 4 bits. Therefore, for 10/100 mode, the RGMII block drives the gmii_txd[3:0] data to the PHY on rgmii_txd_o[3:0] at the rising edge of eth_gmii_gtx_clk.

Receive data conversion

In 1000-Mbps mode, the RGMII block registers the 4-bit data on the rgmii_rxd_i[3:0] signal at both edges of the Receive clock (eth_rx_clk) and transfers the resulting 8-bit data to the MAC GMII on the rising edge of eth_rx_clk clock, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk
- gmii_rxd[7:4] is the value received on rgmii_rxd_i[3:0] at the falling edge of eth_rx_clk

In 10/100 mode, the RGMII block registers the 4-bit data on rgmii_rxd_i[3:0] only at the rising edge of eth_rx_clk and transfers the data to the MAC GMII on the rising edge of eth_rx-_clk, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk.
- gmii_rxd[7:4] is 0x0.

77.7 Ethernet low-power modes

77.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY

layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog and jabber timeout register (ETH_MACWJBTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

Magic packet data format

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (*MAC Address 0 high register (ETH_MACA0HR)* and the *MAC Address 0 low register (ETH_MACA0LR)*) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type…… FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
…CRC

Description of remote wake-up packet mode

This section describes the power saving mode based on remote wake-up packet.

Note: The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog and jabber timeout register (ETH_MACWJBTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The MAC implements a filter lookup table (programmed through *Remote wake-up packet filter register (ETH_MACRWKPFR)* in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in *PMT control status register (ETH_MACPCSR)* and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in *Interrupt status register (ETH_MACISR)* is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in *Interrupt enable register (ETH_MACIER)* is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in *Table 851: Remote wake-up packet filter register*.

Table 851. Remote wake-up packet filter register

ETH_MACRWKPFR value	Field							
0	Filter 0 byte mask							
1	Filter 1 byte mask							
2	Filter 2 byte mask							
3	Filter 3 byte mask							
4	Reserved	Filter 3 command	Reserved	Filter 2 command	Reserved	Filter 1 command	Reserved	Filter 0 command
5	Filter 3 offset		Filter 2 offset		Filter 1 offset		Filter 0 offset	
6	Filter 1 CRC - 16				Filter 0 CRC - 16			
7	Filter 3 CRC - 16				Filter 2 CRC - 16			

The remote wake-up filter fields are described in *Table 852: Description of the remote wake-up filter fields*.

Table 852. Description of the remote wake-up filter fields

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic(1). When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.

Table 852. Description of the remote wake-up filter fields (continued)

Register	Description
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block. – The 16-bit CRC calculation uses the following polynomial: $G(x) = x^{16} + x^{15} + x^2 + 1$ Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

1. The And_Previous bit setting is applicable within a set of four filters.
Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.
If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.
If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.

The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received.

Table 853 lists the remote wake-up scenarios in which PMT interrupt is generated.

Table 853. Remote wake-up packet and PMT interrupt generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode. When the application clears the PWRDWN bit in *Remote wake-up packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the *Channel x transmit control register (ETH_DMACxTXCR)*.
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in *Debug register (ETH_MACDR)* and TXQSTS bit in *Tx queue x debug register (ETH_MTLTXQxDR)* of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in *Operating mode configuration register (ETH_MACCR)*.
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in *Rx queue x debug register (ETH_MTLRXQxDR)* of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the *PMT control status register (ETH_MACPCSR)*.
- Set bit 31 (ARPEN) in the *Operating mode configuration register (ETH_MACCR)*.
- Enable the MAC receiver by setting RE bit and then set PWRDWN bit in the *PMT control status register (ETH_MACPCSR)* to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the *PMT control status register (ETH_MACPCSR)*.
  - Perform a write operation (with reset values) to the *PMT control status register (ETH_MACPCSR)* and the *Remote wake-up packet filter register (ETH_MACRWKPFR)* so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the *Operating mode configuration register (ETH_MACCR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

77.7.2 Energy Efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The

EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps and 1000 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of *LPI timers control register (ETH_MACLTCR)*. The PHY Link Status PLS bit of the *LPI control and status register (ETH_MACLCSR)* indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode. However, the MAC can stop the GTX_CLK clock during the LPI state in the GMII (1000) code.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1,000 Mbps).
4. Updates the status (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the *LPI timers control register (ETH_MACLTCR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Figure 990 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 990) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 991.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 990. LPI transitions (Transmit, 100 Mbds)

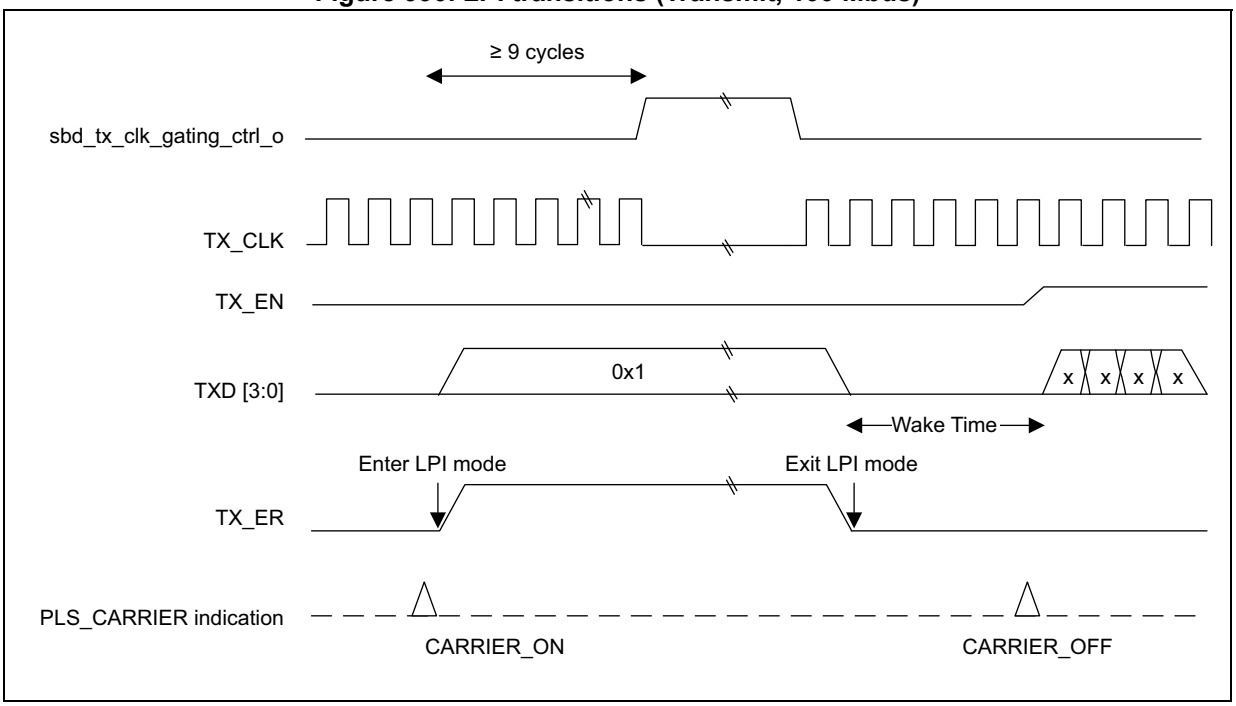
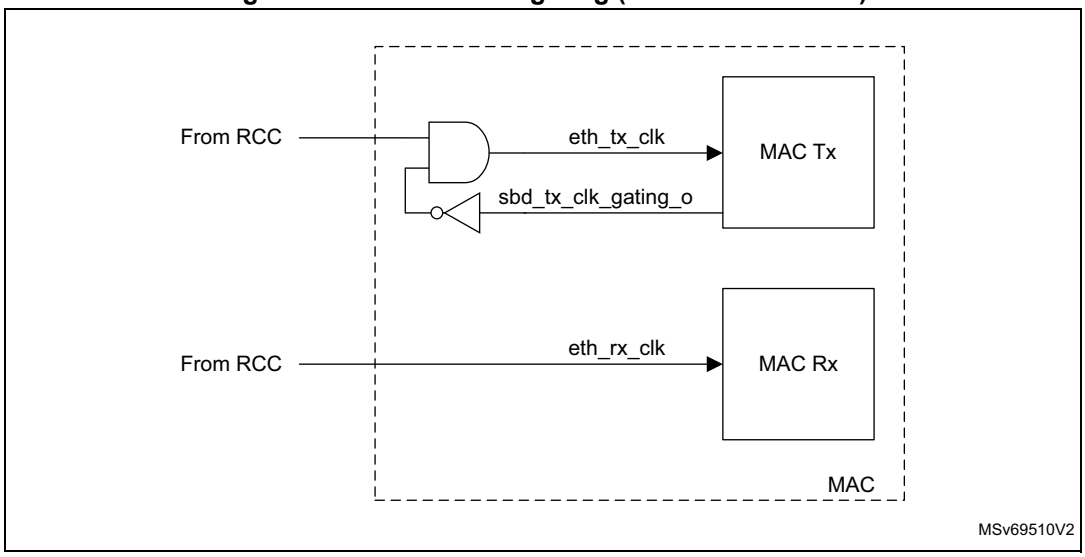


Figure 991. LPI Tx clock gating (when LPITCSE = 1)



Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the *LPI control and status register (ETH_MACLCSR)*.

When LPITXA and LPIEN of *LPI control and status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbps) or RXD[7:0] to 0x01 (for 1,000 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the *LPI control and status register (ETH_MACLCSR)* and immediately generates an interrupt.

Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

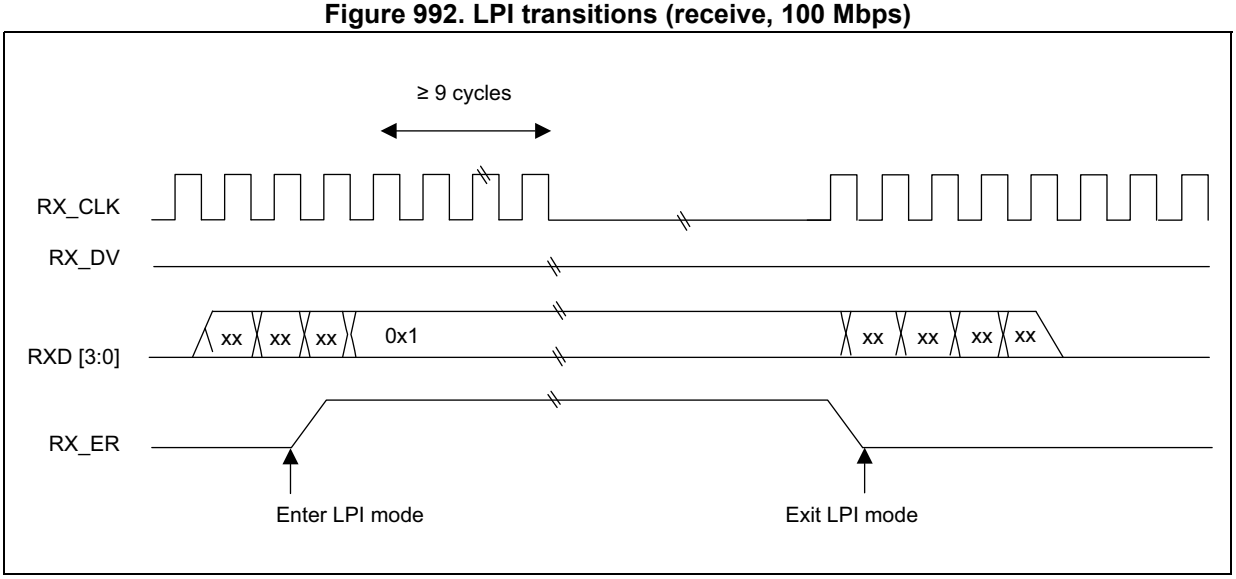
If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the *LPI control and status register (ETH_MACLCSR)* and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 992 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 992. LPI transitions (receive, 100 Mbps)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 992.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the *LPI timers control register (ETH_MACLTCR)* and *LPI entry timer register (ETH_MACLETR)*:

- LPI LS TIMER

  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the *PHYIF control status register (ETH_MACPHYCSR)* by setting the PLSEN bit of *LPI control and status register (ETH_MACLCSR)*.

  The link status is indicated to the MAC by the LNKSTS bit of the *PHYIF control status register (ETH_MACPHYCSR)* or the value programmed by the software in the PLS bit of the *LPI control and status register (ETH_MACLCSR)*. If the link status is not available in the *PHYIF control status register (ETH_MACPHYCSR)*, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.

  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the *LPI timers control register (ETH_MACLTCR)*. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.

- LPI TW TIMER

  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of *LPI timers control register (ETH_MACLTCR)*. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.

  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.

- LPI AUTO ENTRY TIMER

  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in *LPI control and status register (ETH_MACLCSR)* is set.

Note: Program the PLS bit of LPI control and status register (ETH_MACLCSR) to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the Tx clock frequency can result in incorrect timeout.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the *LPI control and status register (ETH_MACLCSR)*.

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the *LPI control and status register (ETH_MACLCSR)* is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see *Section 77.9.13: Programming guidelines for Energy Efficient Ethernet (EEE) on page 4455*.

77.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMACSR: Channel Status (see *Channel x status register (ETH_DMACxSR)*))
- ETH_MTLISR: Interrupt Status (see *Interrupt status register (ETH_MTLISR)*)
- ETH_MACISR: Interrupt Status (see *Interrupt status register (ETH_MACISR)*)

77.8.1 DMA interrupts

Interrupt registers description

The ETH_DMACSR: Channel Status register (see *Channel x status register (ETH_DMACxSR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMACIER: Channel Interrupt Enable register (see *Channel x interrupt enable register (ETH_DMACxIER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMACSR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Periodic scheduling of Transmit and Receive interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 857: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 870: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (ETH_DMACxRXIWTR: Channel i Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in *Table 870: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMACIER register (see *Channel x interrupt enable register (ETH_DMACxIER)*). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (*Channel x status register (ETH_DMACxSR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel x interrupt enable register (ETH_DMACxIER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 854* explains the behavior of the Transfer Complete interrupt.

Table 854. Transfer complete interrupt behavior

Interrupt mode	Behavior of TI/RI and eth_sbd_intr_it
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. The bits get cleared whenever the software driver writes 1 to these bits. The eth_sbd_intr_it is asserted whenever the corresponding interrupts are also enabled in ETH_DMACxIER register.
INTM=1	The TI/RI is set as explained above. However, the eth_sbd_intr_it is not asserted for any RI/TI events.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another Transfer Complete event is detected before it is cleared (serviced) by the software, these status bits is automatically set again. However, the eth_sbd_intr_it is not generated based on TI/RI.

77.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQ0ICSR: Queue 0 Interrupt Control Status or ETH_MTLQ1ICSR: Queue 1 Interrupt Control Status must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events on Queue 0 and Queue 1:

- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

77.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  - MMC Receive Checksum Offload Interrupt Status
  - MMC transmit interrupt status
  - MMC Receive interrupt status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt
- RGMII Interrupt Status

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.

77.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 77.9.1*)
- MTL initialization (see *Section 77.9.2*)
- MAC initialization (see *Section 77.9.3*)
- Performing normal receive and transmit operation (see *Section 77.9.4*)
- Stopping and starting transmission (see *Section 77.9.5*)
- Programming guidelines for multichannel multiqueue operation (see *Section 77.9.8*)
- Programming guidelines for GMII link state transitions (see *Section 77.9.9*)
- Programming guidelines for IEEE 1588 timestamping (see *Section 77.9.10*)
- Programming guidelines for AV feature (see *Section 77.9.12*)
- Programming guidelines for Energy Efficient Ethernet (see *Section 77.9.13*)
- Programming guidelines for flexible pulse-per-second (PPS) output (see *Section 77.9.14*)
- Programming guidelines for IEEE 1588 auxiliary snapshot (see *Section 77.9.15*)
- Programming guidelines for TSO (see *Section 77.9.16*)
- Programming guidelines for VLAN filtering on the receiver (see *Section 77.9.17*)
- Programming guidelines for extended VLAN filtering and routing on reception (see *Section 77.9.18*)
- Programming guidelines for queue/channel-based VLAN inclusion register (see *Section 77.9.19*)
- Programming guidelines for EST (see *Section 77.9.20*)
- Programming the launch time in time-based scheduling (see *Section 77.9.21*)
- Enabling the frame preemption function (see *Section 77.9.22*)

77.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL, ONEKBBE, RD_OSR_LMT, and WR_OSR_LMT
   b) Fixed burst or undefined burst
   c) Outstanding requests limit value (OSR_LMT).
   d) If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
   e) EAME.
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 77.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMACTXRLR and ETH_DMACRXRLR registers (see *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).

Note: In 40-bit addressing mode, program the higher address List registers in Channel x Tx descriptor list high address register (ETH_DMACxTXDLHAR) and (Channel x Rx descriptor list high address register (ETH_DMACxRXDLHAR).

The tail pointer registers must be advanced to the location immediately after the descriptors that are set, for the DMA to know that additional descriptors are available.

7. Program ETH_DMACCR, ETH_DMACTXCR and ETH_DMACRXCR registers (see *Channel x control register (ETH_DMACxCR)*, *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel x receive control register (ETH_DMACxRXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.
8. Enable the interrupts by programming the ETH_DMACIER register (see *Channel x interrupt enable register (ETH_DMACxIER)*).
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of *Channel x receive control register (ETH_DMACxRXCR)* and ST (bit 0) of the ETH_DMACTXCR (see *Channel x transmit control register (ETH_DMACxTXCR)*).
10. Repeat steps 4 to 9 for all the Tx DMA channels selected in the hardware.

77.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in *Operating mode register (ETH_MTLOMR)* to initialize the MTL operation when multiple Tx and Rx queues are used.
2. Program the Receive Queue to DMA mapping in *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)*.
3. Program the following fields to initialize the operating mode in *Tx queue x operating mode register (ETH_MTLTXQxOMR)*.
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).
4. Program the following fields to initialize the operating mode in the ETH_MTLRXQxOMR register (see *Rx queue x operating mode register (ETH_MTLRXQxOMR)*):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).
5. Repeat the previous two steps for all the MTL Tx and Rx queues selected in the configuration

77.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *MAC Address x low register (ETH_MACAxLR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x high register (ETH_MACAxHR)*.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*:
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.

4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register. For example, Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to 16383.

77.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMACTXDTPR (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and ETH_DMACRXDTPR (see *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).
4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel x current application receive descriptor register (ETH_DMACxCARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel x current application receive descriptor register (ETH_DMACxCARXDR)*).

77.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMACTXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)*, PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.

Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.

77.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

77.9.7 Programming guidelines for switching the AXI clock frequency

To dynamically change the AXI clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the *Tx queue x*

debug register (ETH_MTLTXQxDR) and the status of DMA is given in *Debug status 1 register (ETH_DMADS1R)*.

2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in *Operating mode configuration register (ETH_MACCR)*.
3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in *Rx queue x debug register (ETH_MTLRXQxDR)*.
4. Ensure that the application does not perform any register read or write operation.
5. Change the frequency of the AXI clock.
6. Enable the MAC transmitter or the MAC receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

77.9.8 Programming guidelines for multichannel multiqueue operation

Transmit path

1. Program the Transmit queue size in the TQS field in ETH_MTLTXQ**x**OMR register. The size of the queue is determined by the value programmed in the TQS field.

   During the Transmit operation, the number of channels is equal to the number of the queues. As a result, the channel to queue mapping is fixed.
2. To use a queue, the queue must be enabled by setting the TXQEN bit in the corresponding ETH_MTLTXQ**x**OMR register. The ST bit in *Channel x transmit control register (ETH_DMACxTXCR)* and the corresponding TXQEN bit in ETH_MTLTXQ**x**OMR register must be enabled.
3. Program the scheduling method in SCHALG bit of ETH_MTLOMR register.
4. If CBS algorithm is enabled on AV queue 1, ETH_MTLTXQ1ECR, ETH_MTLTXQ1SSCR**,** ETH_MTLTXQ1HCR and ETH_MTLTXQ1LCR registers also need to be programmed as required.

Receive path

1. Program the Receive queue size in the RQS field in ETH_MTLRXQ**x**OMR register. The size of the queue is determined by the value programmed in the RQS field.
2. Enable the Receive queues 0 to 1 in the RXQ0EN to RXQ1EN fields of ETH_MACRXQ0CR register for AV.

   Enable SR bit of statically or dynamically mapped *Channel x receive control register (ETH_DMACxRXCR)* and corresponding RXQiEN in *Rx queue control 0 register (ETH_MACRXQC0R)*.
3. The MAC routes the Rx packets to the Rx queues depending on following packet types:
   a) AV PTP packets are routed according to the value of PTPQ in ETH_MACRXQC1R register
   b) AV untagged control packets are routed according to the value of AVCPQ in ETH_MACRXQC1R register.
   c) VLAN tag priority field in VLAN tagged packets

      Program PSRQ1 and PSRQ0 of the ETH_MACRXQC2R register to configure the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 0 to 1.

d) The AV tagged control and data packets are also routed according to PSRQ field of the ETH_MACRXQC2R register.

Note: The priorities set in PSRQ0 and PSRQ1 must be unique.

4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
   a) Program the RAA field of *Operating mode register (ETH_MTLOMR)* to select the arbitration algorithm to decide which Rx queue is read out from the RxFIFO memory.
   b) Program the *Rx queue x control register (ETH_MTLRXQxCR)* to decide the weights and the packet arbitration for each Rx queue.
   c) If static mapping is programmed in Rx queue and *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)* (QiDDMACH is reset to 0), bits QiMDMACH need to be programmed to select the channel for which each queue is mapped.
   d) Set RXQiDDMACH bit in Rx queue and *Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)* to select dynamic mapping of packets in each Rx queue.
   e) In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC address register.

Programming guidelines for recovering from DMA channel failure

When the DMA channel issues a bus error, follow the steps described below to recover from the failure.

Recovering from the receive DMA channel failure

Follow these steps if a bus error occurred in the Receive DMA channel:

1. Set the RPF bit to 1. This flushes all the packets one after the other. This step is optional. However, setting this bit prevents HOL (head-of-line) blocking in the Rx queues when packets sent to the RXDMA are stopped due to bus error.
2. Reprogram the specific registers of the DMA channel.
3. Start the DMA channel.

Recovering from the transmit DMA channel failure

1. Stop the specific DMA channel, even if it is in active state.
2. Flush the corresponding MTL queue.
3. Reprogram the specific registers of the DMA channel.
4. Start the DMA channel.

Note: Due to the known limitations, reprogramming the DMA channel registers might not always be successful to recover from a bus error. If the peripheral is not fully functional after reprogramming the DMA, as a workaround, issue a soft reset to recover from the bus error.

Programming guidelines for disabling receive queue

Follow these steps to disable specific receive queue:

1. Disable the receiver by setting the RE bit of the *Operating mode configuration register (ETH_MACCR)* to 0.
2. Read the *Debug register (ETH_MACDR)* (ensure that RPESTS and RFCFCSTS fields are 0) and *Rx queue x debug register (ETH_MTLRXQxDR)* of all the Rx queues.

3. When all *Rx queue x debug register (ETH_MTLRXQxDR)* are read as 0, disable the intended Rx queue by programming the corresponding fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.
4. Enable the receiver by setting the RE bit of *Operating mode configuration register (ETH_MACCR)* to 1.

77.9.9 Programming guidelines for GMII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel x control register (ETH_DMACxCR)*.
2. Disable the MAC receiver by clearing RE bit of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing TE bit of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] to 00 in *Rx queue x debug register (ETH_MTLRXQxDR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC transmitter and receiver.

   The Rx DMA does not need to be enabled: since the receiver is disabled, there are no data in the Rx FIFO.

Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Disable the MAC transmitter and receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

77.9.10 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control Register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control Register (ETH_MACTSCR)* to enable timestamping.
3. Program *Subsecond increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control Register (ETH_MACTSCR)*.
5. Poll the *Timestamp control Register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control Register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control Register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control Register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control Register (ETH_MACTSCR)*.

   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend to increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control Register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 3 in Register *Timestamp control Register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

77.9.11 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Sync message.
5. Program the LSI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Sync messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Sync message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.
3. Program the APDREQEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Pdelay_Req message.
5. Program the LMPDRI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Pdelay_Req messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in *Log message interval register (ETH_MACLMIR)*. The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
4. Program the DRSYNCR and LMPDRI fields in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the DRSYNCR, LSI and LMPDRI fields in *Log message interval register (ETH_MACLMIR)*, the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 3, X, and X respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP*

Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

4. Program the LMPDRI field in *Log message interval register (ETH_MACLMIR)* to indicate periodicity of the PTP Pdelay_Req messages
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

77.9.12 Programming guidelines for AV feature

Initializing the DMA

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all registers and logic (bit 0 in ETH_DMAMR).
2. Wait for the completion of the reset process. Poll bit 0 of the ETH_DMAMR, which is cleared only when the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in ETH_DMAMR register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the Transmit and Receive descriptors. When OSF mode is used, at least two TX descriptors are required. For more information about descriptors, refer to *Section 77.10: Descriptors*.
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the Transmit and Receive Ring length registers (*Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*). The ring length programmed must be at least 4.
7. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*). In addition, program the Transmit and Receive tail pointer registers indicating the available descriptors to the DMA (*Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).
8. Program the following fields to initialize the operating mode in the ETH_MTLTXQ0OMR:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 10 to enable Transmit Queue0
   d) Transmit Queue Size (TQS)
9. Enable the interrupts by programming the EH_DMAC0IER.
10. Repeat steps 4 to 9 for all the Tx DMA and RX DMA channels.

11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues.
12. Start the Receive and Transmit DMA by setting bit 0 of the ETH_DMAC0TXCR and bit 0 of the ETH_DMAC0RXCR (see *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel x receive control register (ETH_DMACxRXCR)*).

Enabling slot number checking

You can use the slot number check feature to specify the intervals at which the DMA Channels mapped to AV queues fetches the frames from the AHB/AXI system bus. This feature is useful to ensure a uniform and periodic transfer of the AV traffic from the host memory. It is available only when timestamping is enabled and the *Subsecond increment register (ETH_MACSSIR)* is programmed. Complete the following steps to enable the slot number checking:

Note: The slot number checking steps should be complete after programming the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues (step 11 of Section : Initializing the DMA). and before starting the Receive and Transmit DMA (step 12).

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Make sure that the SLOTNUM field (bits 22 to 19) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. To do this, read the current reference slot number from the *Channel x slot function control status register (ETH_DMACxSFCSR)*.
3. Set bit 0 (ESC) of *Channel x slot function control status register (ETH_DMACxSFCSR)* to enable the slot number checking.

Enabling average bits per slot reporting

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Program SLC bits of the *Tx queue x ETS control register (ETH_MTLTXQxECR)* of a channel with number of slots over which the average transmitted bits per slot need to be computed.
3. Enable ABPSIE of the *Queue x interrupt control status register (ETH_MTLQxICSR)* of a channel to generate the average bits per slot interrupt.

Note: The frequency of this interrupt depends on the value programmed during previous step. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.

When it is not required, you can disable this interrupt to stop the interrupt flooding.

4. Read ABS bits from *Tx queue x ETS status register (ETH_MTLTXQxESR)* of a channel on each interrupt.

   The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 1 (ABPSIS) of *Queue x interrupt control status register (ETH_MTLQxICSR)* indicates that a new value is updated in the ABS field.

Disabling flow control for AV-enabled queues

Transmit flow

Program the EHFC (Enable Hardware Flow Control) bit of corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* to 0.

77.9.13 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 15 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control and status register (ETH_MACLCSR)*.

   Update *LPI control and status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program *One-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
6. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of *LPI control and status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
8. Set bit 16 of *LPI control and status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
10. The MAC transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC transmitter from LPI state.

Note: To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the GMII Transmit clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal to gate the clock input.

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when *LPI control and status register (ETH_MACLCSR)* is read.

Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* to exit the MAC from LPI mode.

77.9.14 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS x*

target time seconds register (ETH_MACPPSTTSxR) and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) as start time of PPS signal output.

2. Program the start time value in the Target Time registers (register *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).
3. Program the width of the PPS signal output in *PPS x width register (ETH_MACPPSWxR)* Register.
4. Program PPSCMD[3:0] of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS x width register (ETH_MACPPSWxR)*.
3. Program PPSCMD[3:0] bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

   If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS x*

target time seconds register (ETH_MACPPSTTSxR) and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) for start time of the PPS signal output.

2. Program the start time value in the Target Time registers (register *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS x interval register (ETH_MACPPSIxR)*.
4. Program the width of the PPS signal output in *PPS x width register (ETH_MACPPSWxR)*.
5. Program PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to

the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupt, the correct writing order is as follow:

1. *PPS x target time nanoseconds register (ETH_MACPPSTTNxR).*
2. *PPS x target time seconds register (ETH_MACPPSTTSxR).*
3. *PPS x interval register (ETH_MACPPSIxR).*
4. *PPS x width register (ETH_MACPPSWxR).*
5. *PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).*

77.9.15 Programming guidelines for IEEE 1588 auxiliary snapshot

The IEEE 1588 auxiliary snapshot function is used to capture the auxiliary snapshot for a trigger event received on the ptp_aux_ts_trig_i input.

To program the IEEE 1588 auxiliary snapshot function, complete the following steps:

1. Program the TSIE bit in the *Interrupt enable register (ETH_MACIER)* to enable interrupt generation when the auxiliary snapshot is captured in the FIFO.
2. Program the ATSENx bit of the *Auxiliary control register (ETH_MACACR)* to enable the capture of the auxiliary snapshot based on the respective trigger input.
3. If the software does not want the captured auxiliary snapshot to be read, program the ATSFC bit of *Auxiliary control register (ETH_MACACR)* anytime to clear the auxiliary snapshot FIFO.
4. When TSIS is set or an interrupt is generated (due to TSIE programmed to 1), read the *Timestamp status register (ETH_MACTSSR)*, *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)* to access the captured auxiliary snapshot.

Note: When the auxiliary snapshot is captured in the FIFO for a trigger event, the following bitfields are updated:

- *AUXTSTRIG (status bit indicating that an auxiliary snapshot has been captured), ATSSTN (trigger event input source identifying the first available entry in the FIFO), and ATSNS (indicating the number of auxiliary snapshots available in the FIFO) bitfields of the Timestamp status register (ETH_MACTSSR).*
- *TSIS (indicating that an interesting event occurred in the IEEE 1588 timestamp function) bitfield in the Interrupt status register (ETH_DMAISR).*

The AUXTSTRIG bit is cleared on read (or set when the RCWE bit of the CSR software control register (ETH_MACCSRSWCR) is set).

Note: When the captured auxiliary snapshot is ignored because that the FIFO is full, the ATSSTM bit, which indicates that a trigger has been missed, is set.

77.9.16 Programming guidelines for TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding *Channel x transmit control register (ETH_DMACxTXCR)* to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS[13:0] of *Channel x control register (ETH_DMACxCR)*
   – or MSS in the context descriptor

   If MSS[13:0] field is programmed in both *Channel x control register (ETH_DMACxCR)* and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

77.9.17 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag Control register (ETH_MACVTCR)* for the following bit to select the filtering method:
   – ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   – VTHM: VLAN Tag Hash Table Match Enable.
   – ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   – ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   – DOVLTC: Ignores VLAN Type for Tag Match
   – VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag Control register (ETH_MACVTCR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*.
   – When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the *VLAN Hash table register (ETH_MACVHTR)*.
   – When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of *VLAN Hash table register (ETH_MACVHTR)*.

   For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

77.9.18 Programming guidelines for extended VLAN filtering and routing on reception

For the indirect access to the per VLAN Tag registers, follow these steps:

- Write access
  a) Write the required data into the *VLAN tag data register (ETH_MACVTDR)*.
  b) Program the OFS field of *VLAN tag Control register (ETH_MACVTCR)* with the required filter register offset and command type to the CT field. For a write command, set this bit to 0.
  c) Write 1 to the OB field and wait until the OB bit is reset before performing the next write. This is to ensure that the appropriate VLAN Tag Filter Register has been programmed.
- Read access
  a) Program the OFS field of the *VLAN tag Control register (ETH_MACVTCR)* with the required register offset and command type to the CT field. For a read command, set this bit to 1.
  b) Write 1 to the OB field and wait till the OB bit is reset. The appropriate VLAN Tag Filter register value is available in the *VLAN tag data register (ETH_MACVTDR)*.

77.9.19 Programming sequence for queue/channel-based VLAN inclusion register

Writing to the per queue/channel VLAN inclusion indirect registers

To write to the queue/channel-based VLAN inclusion register, complete the following steps.

Note: Indirect VLAN include registers are not accessed while setting the CBTI bit.

1. Set the CBTI bit of *VLAN inclusion register (ETH_MACVIR)* to 1, to enable queue/channel based VLAN tag insertion on all transmitted packets. This bit must be set before any indirect access to the queue/channel specific *VLAN inclusion register [alternate] (ETH_MACVIR)*.
2. Set the RDWR field of *VLAN inclusion register (ETH_MACVIR)* to 1, to indicate write access.
3. Program the VLAN tag and VLAN type to be inserted in the packets from a particular queue/channel in VLT[15:0] and CSVL fields of *VLAN inclusion register (ETH_MACVIR)*. The corresponding offset address in ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set. Set the RDWR bit to 1 to indicate write access. The write to byte 0 of *VLAN inclusion register (ETH_MACVIR)* initiates access to indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*.
4. The BUSY bit of *VLAN inclusion register (ETH_MACVIR)* is set by the peripheral to indicate the progress of access to indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*. On completion of the access, the BUSY bit is cleared. The application must not attempt subsequent access to *VLAN inclusion register [alternate] (ETH_MACVIR)* when the BUSY bit is 1.
5. Repeat step 2 and step 3 to program VLAN tag and VLAN type to be inserted in packets from the remaining queues/channels. The application must ensure that the required VLAN tag and VLAN type for all the queues/channels are programmed; otherwise unintended VLAN tag and VLAN type might be inserted.

Read the per queue/channel VLAN inclusion indirect registers

To read the queue/channel-based VLAN inclusion register, complete the following steps:

1. Set the CBTI field of *VLAN inclusion register (ETH_MACVIR)* to 1, to enable queue/channel based VLAN tag insertion on all the transmitted packets. This bit must be set before indirect access to the queue/channel specific *VLAN inclusion register [alternate] (ETH_MACVIR)*.
2. Program the read offset address in the ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on).

   Set the RDWR bit to 0 to indicate read access.
3. The BUSY field of *VLAN inclusion register (ETH_MACVIR)* is set by the peripheral to indicate the progress of access to the indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*. On completion of the access, the BUSY field is cleared. Bits[15:0] and bit [19] of *VLAN inclusion register (ETH_MACVIR)* contain the VLAN tag and VLAN type respectively, from the corresponding queue/channel/offset address.
4. Repeat step 2 and step 3 to read VLAN tag and VLAN type from the remaining queues/channels.

77.9.20 Programming guidelines for EST

Program the gate control values and time intervals in the software owned gate control list (SWOL) along with the other EST related registers that are described in *Section 77.11.3: Ethernet MTL registers*, to appropriate values. The following subsections provide step-by-step details for programming the GCL and the other EST related registers.

Programming the GCL and GCL-linked registers

Follow these steps to program the gate control list (GCL) and the four other registers implemented per GCL.

1. The GCL and the four other GCL-linked registers should be accessed through indirect addressing using the *EST Gate Control List register (ETH_MTLESTGCLCR)* and *EST Gate Control List Data register (ETH_MTLESTGCLDR)*. The SWOL bitfield of the *EST Status register (ETH_MTLESTSR)* indicates if GCL0 or GCL1 is owned by software.
2. To program the GCL, write the 32-bit write data to *EST Gate Control List Data register (ETH_MTLESTGCLDR)*. Then program *EST Gate Control List Data register (ETH_MTLESTGCLDR)* register to write the write address and other control information.
3. In the *EST Gate Control List Data register (ETH_MTLESTGCLDR)*, writing data consists of 8 bits of gate controls plus 24 bits of time interval. Gate close is indicated by

programming a 0 and gate open is indicated by programming a 1. The data should be written in the following format:

{TC7, TC6, TC5, TC4, TC3, TC2, TC1, TC0, 24-bit time interval} where TCx = 0 or 1.

4. In the *EST Gate Control List Data register (ETH_MTLESTGCLDR)*, program the SRWO bit to 1 (to start a write operation) and program the address and r/w bitfields appropriately.
5. Poll and check for the SRWO bit to be cleared by hardware to indicate the completion of the previous operation before initiating a new read/write operation using the same indirect addressing mode.
6. Repeat steps 3, 4, 5 until the programming of the GCL is complete.
7. Using the indirect addressing method, program the BTR, CTR, TER and LLR registers. Set the GCRR bit in the *EST Gate Control List Data register (ETH_MTLESTGCLDR)* appropriately. The GCRR bit interprets the address field as belonging to these registers (instead of the GCL).

After programming the GCL and the related registers, program the *EST Control register (ETH_MTLESTCR)* to allow the hardware to own and process the GCL. When the List length (as indicated in LLR) is 1, the associated time interval should be smaller than the value of the cycle time register. Otherwise, an error is reported (as detailed in the Error handling section) as a single set of gate controls adds no value in the TSN context.

Note: The time unit in all the GCL related registers is seconds and nanoseconds. In cases where internally generated PTP system time is used, the nanoseconds field must be programmed to use the Digital rollover mode (TSCTRLSSR bitfield of Timestamp control Register (ETH_MACTSCR) set to 1).

Programming the EST registers

After completing the steps mentioned in *Section : Programming the GCL and GCL-linked registers*, program the *EST Control register (ETH_MTLESTCR)*.

1. Set the current time offset value and time internal left shift bitfields of the *EST Control register (ETH_MTLESTCR)* appropriately. In addition, set the enable EST (EEST) and switch to SWOL (SWOL) bits.
2. This enables the Ethernet controller to own and process the new GCL and switch to the new GCL at the BTR value. If enabled, the Ethernet controller generates an interrupt when the switch to the new list occurs.
3. Software must address any other interrupts received during the hardware execution of the GCL.

Set the SSWL field in the *EST Control register (ETH_MTLESTCR)* to hand off to the controller.

The SSWL bit is reset/cleared by the controller when it successfully switches to the new list. The controller also flips the SWOL bit to indicate the new GCL that the software owns.

To install a new GCL, program the GCL it owns (indicated by SWOL bit) as described in *Section : Programming the GCL and GCL-linked registers* and then program the *EST Control register (ETH_MTLESTCR)* as described above. Ensure that the new BTR is set to an appropriate value to avoid BTR error that might need software intervention in some cases.

To avoid transmission overruns, the packet length (frame size) information should be available at all times. Therefore, in the DMA configurations, program the packet length in the

first descriptor of every Tx frame. Similarly, in the MTL configuration, provide the packet length in the control word.

77.9.21 Programming the launch time in time-based scheduling

The launch time is programmed in the enhanced normal transmit descriptors.

The OSTC and launch time features are mutually exclusive and must not be used together; in case of a conflict (if OSTC = LTV = 1 in MTL configuration), priority is given to OSTC and the launch time is ignored.

If a context descriptor is received with a valid OSTC value immediately before receiving a first normal descriptor with LTV bit set, then the LTV is ignored.

77.9.22 Enabling the frame preemption function

Receive-side programming

1. Enable at least two Rx queues by programming the appropriate values in the *Rx queue control 0 register (ETH_MACRXQC0R)*; one queue for express and the other for preemption when preemption receive traffic is expected.
2. Program the *Rx queue control 1 register (ETH_MACRXQC1R)*, *Rx queue control 2 register (ETH_MACRXQC2R)* and *Rx Queue control register (ETH_MACRXQCR)* to route the received packets to MTL Rx queues. Ensure that the preemptable traffic and express traffic are not routed to the same Rx queue.

   As the queue mapping for tagged packets is based on VLAN user-priority field, the priority of preemptable and express packets must be exclusive to each other. In other words, packets of a certain priority (traffic class) are either express or preemptable but cannot be both.
3. Program the preemption residue queue (FPRQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*) to route all non-classified received preemptive packets, so that these packets are not routed to the default or residue queue for express packets, which is Rx Queue 0.

Transmit-side programming

1. Enable at least two Tx queues by programming appropriate values in the TXQEN[1:0] bitfield of the *Tx queue x operating mode register (ETH_MTLTXQxOMR)*, one for express and the other for preemption (by setting the corresponding bit in the PEC bitfield of *FPE Frame Preemption Control Status register (ETH_MTLFPECSR)*) when preemption receive traffic is expected.
2. Set the EFPE bit of the *FPE control and status register (ETH_MACFPECSR)* to enable the preemption function in the transmitter.
3. Program the AFSZ[1:0] bitfield of the *FPE Frame Preemption Control Status register (ETH_MTLFPECSR)* to set the minimum size of the non-final fragments.
4. When 802.1Qbv/EST (Enhancement to Scheduled Traffic) is also enabled, follow the programming guidelines for EST and GCL.
5. Program the Hold Advance (HADV[15:0]) and Release Advance (RADV[15:0]) bitfields of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)* with how much early the hold and release, respectively, must be initiated, so that the hold and release happen at exactly the boundaries of GCL rows. The value of HADV[15:0] must be

equal to the time required to transmit minimum_fragment_size + IPG/EIPG + Preamble (in nanosecond), so that the packet can be preempted before the hold window starts.

6. Set the LBHT field of *FPE Frame Preemption Control Status register (ETH_MTLFPECSR)* if the hold operation must start from the second row of new GCL without a 0-to-1 transition on the Hold or the Release bit in GCL, that is, Hold or Release bits of GCL row 1 and 2 are set.

77.10 Descriptors

77.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**

  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**

  The context descriptors are used to provide control information applicable to the packet to be transmitted.

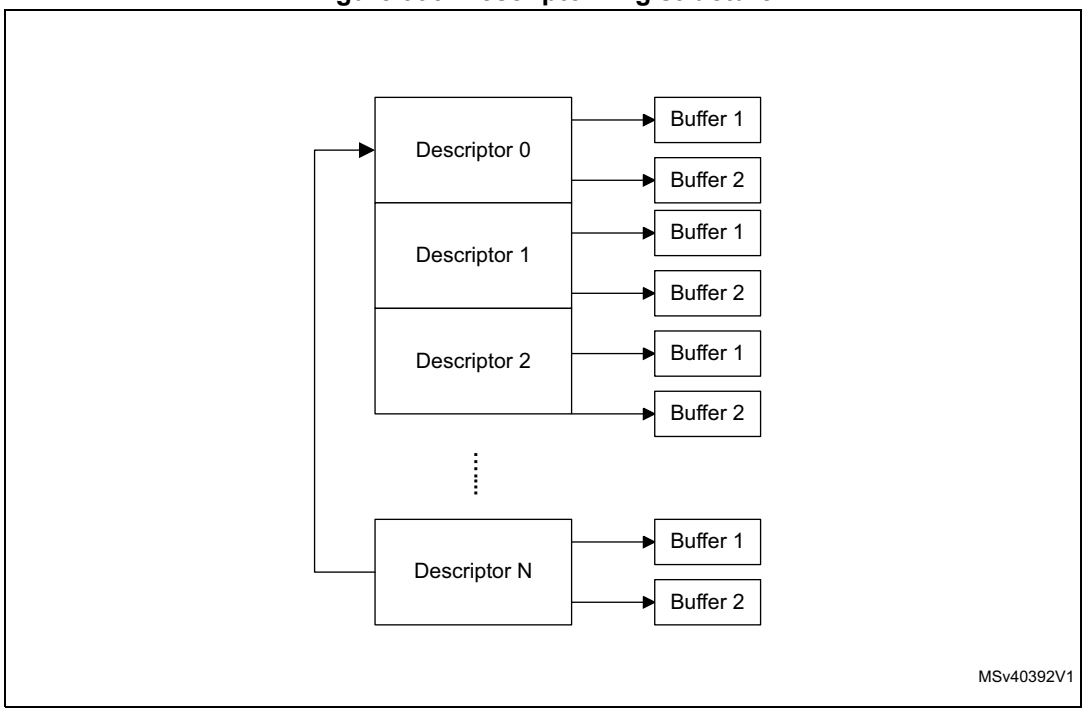
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

77.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 993. Descriptor ring structure



In a ring structure, descriptors are separated by the 64-bit word number programmed in the DSL field of the *Channel x control register (ETH_DMACxCR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)*)
- *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*)

The *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* or *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

```
Descriptor Tail Pointer > Current Descriptor Pointer;
```

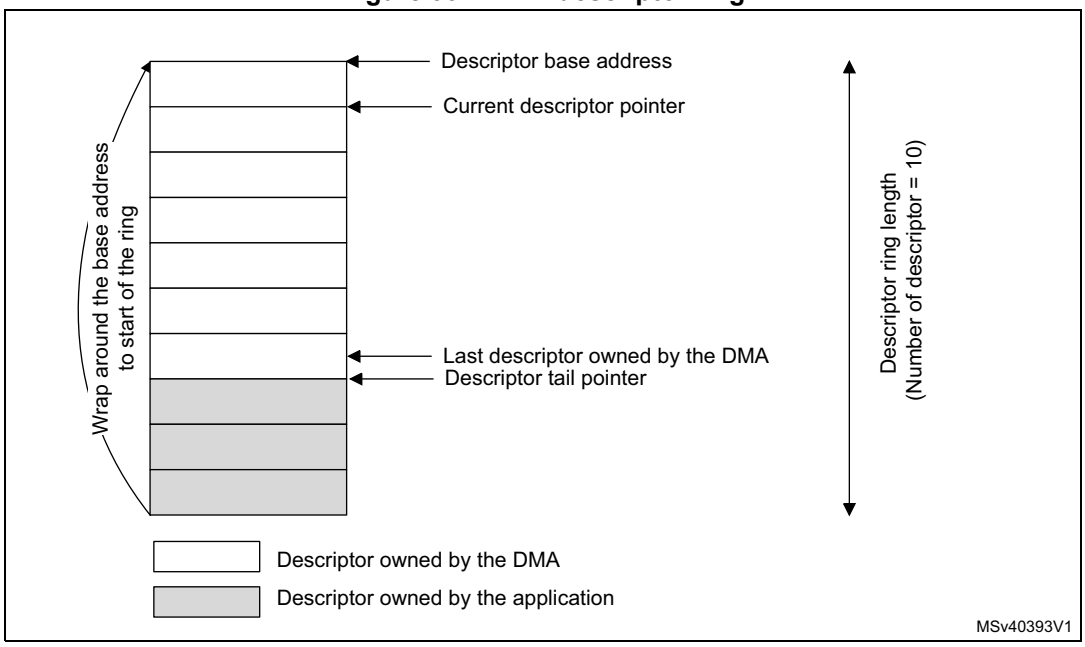
Note: When the application sets the descriptors into the end-of-descriptor ring, continues setting from the start of the descriptor list and updating the descriptor tail pointer, the above condition is not satisfied. The DMA detects such a case, processes the descriptors until the

end of the ring and then continues processing from the start of the descriptor list until the current descriptor pointer is equal to the descriptor tail pointer.

When the application programs the descriptor tail pointer beyond the end of the descriptor ring, the DMA interprets this as the availability of one more complete ring of descriptors, and after processing the last descriptor in the ring continues to process descriptors from the start of the descriptor list. If this is not intended, the application must program the descriptor tail pointer to the start of the descriptor list after setting the last descriptor in the ring.

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 994: DMA descriptor ring*.

Figure 994. DMA descriptor ring



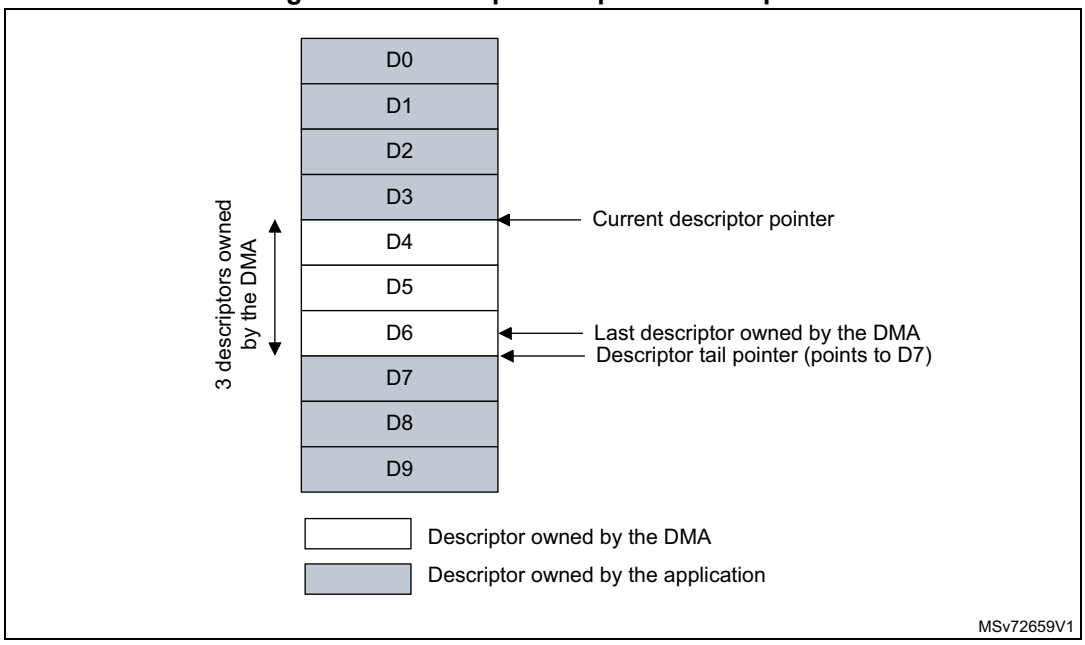
For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

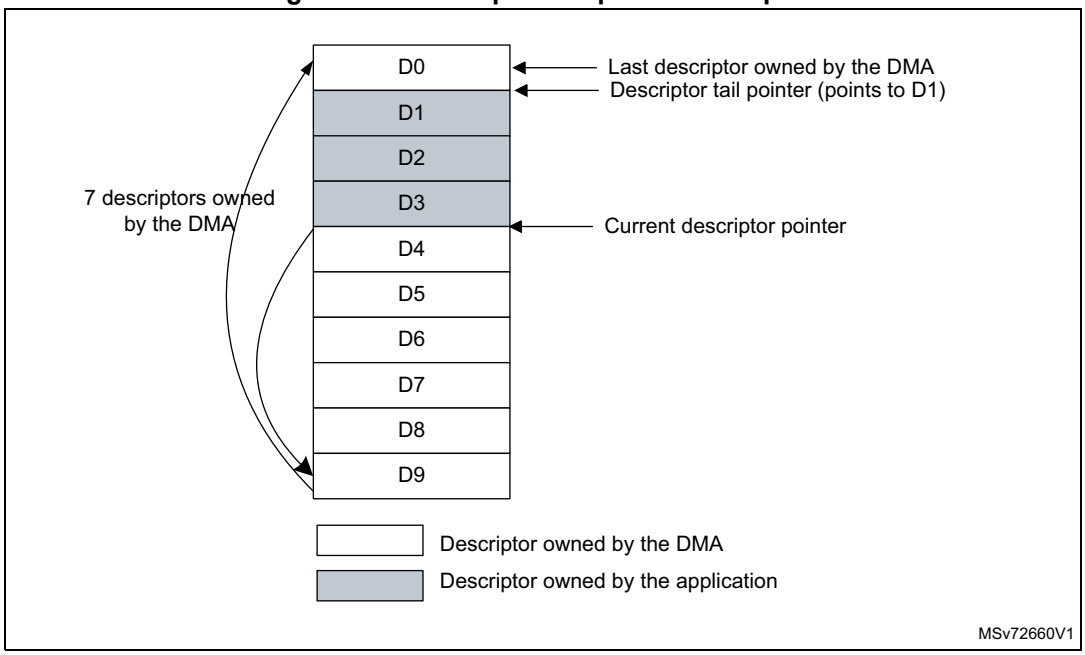
Descriptor tail pointer handling examples

Figure 995. Descriptor tail pointer example 1



When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in *Figure 995: Descriptor tail pointer example 1*.

Figure 996. Descriptor tail pointer example 2



As shown in *Figure 996: Descriptor tail pointer example 2*, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor 1 (D1).

The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

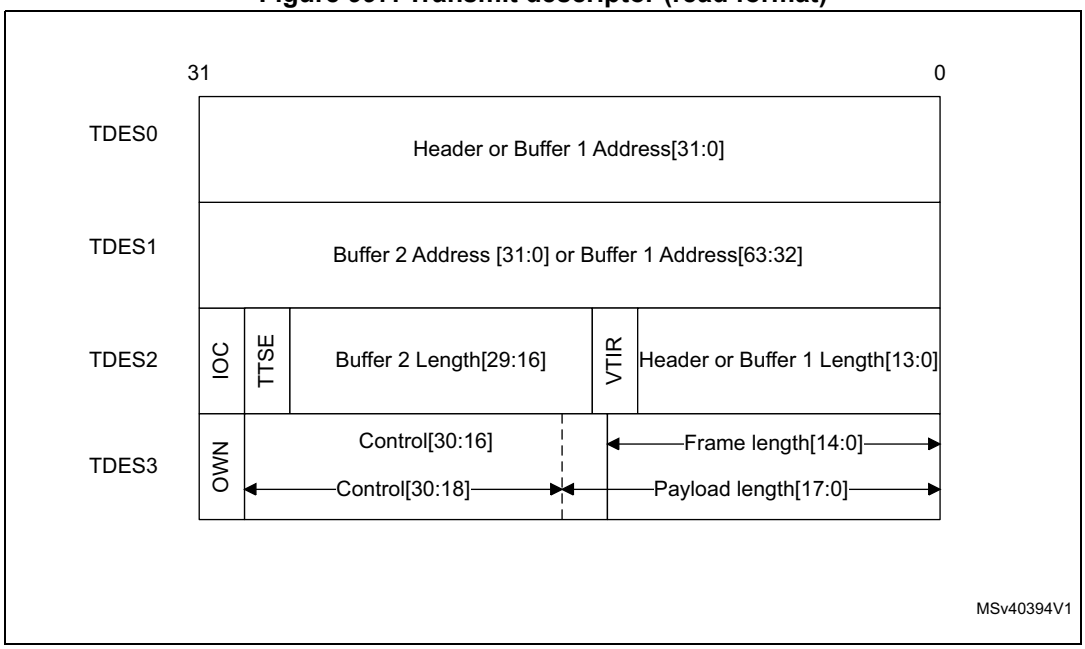
77.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format

Transmit normal descriptor (read format)

Figure 997 shows the Read format for Transmit normal descriptor. *Table 855* to *Table 858* provide a detailed description of all Transmit normal descriptors (read format).

Figure 997. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 855. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer or TSO Header Address Pointer** These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

- TDES1 normal descriptor (read format)

Table 856. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or Buffer 1 Address Pointer:** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment. In 40-bit addressing mode, these bits indicate the most-significant bits of the Buffer 1 Address Pointer.

- TDES2 normal descriptor (read format)

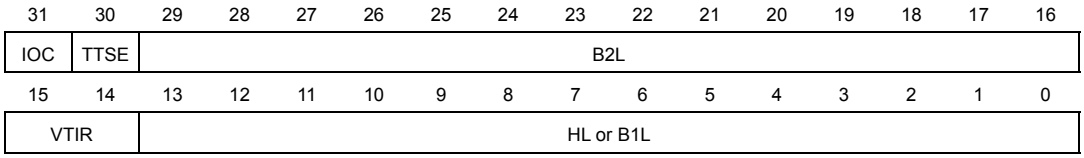


Table 857. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on completion:** This bit sets the TI bit in the *Channel x status register (ETH_DMACxSR)*) when the present packet transmission is complete.
30	TTSE	**Transmit Timestamp Enable** This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor.
29:16	B2L	**Buffer 2 Length** The driver sets this field. When set, this field indicates Buffer 2 length.
15:14	VTIR	**VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: 00: Do not add a VLAN tag. 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.

Table 857. TDES2 normal descriptor (read format) (continued)

Bits	Name	Description
13:0	HL or B1L	**Header length or buffer 1 length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

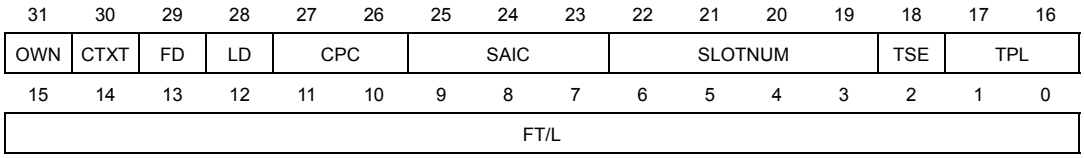


Table 858. TDES3 normal descriptor (read format)

Bits	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.

Table 858. TDES3 normal descriptor (read format) (continued)

Bits	Name	Description
27:26	CPC	**CRC Pad Control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: 00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	**SA Insertion Control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]: 00: Do not include the source address 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.

Table 858. TDES3 normal descriptor (read format) (continued)

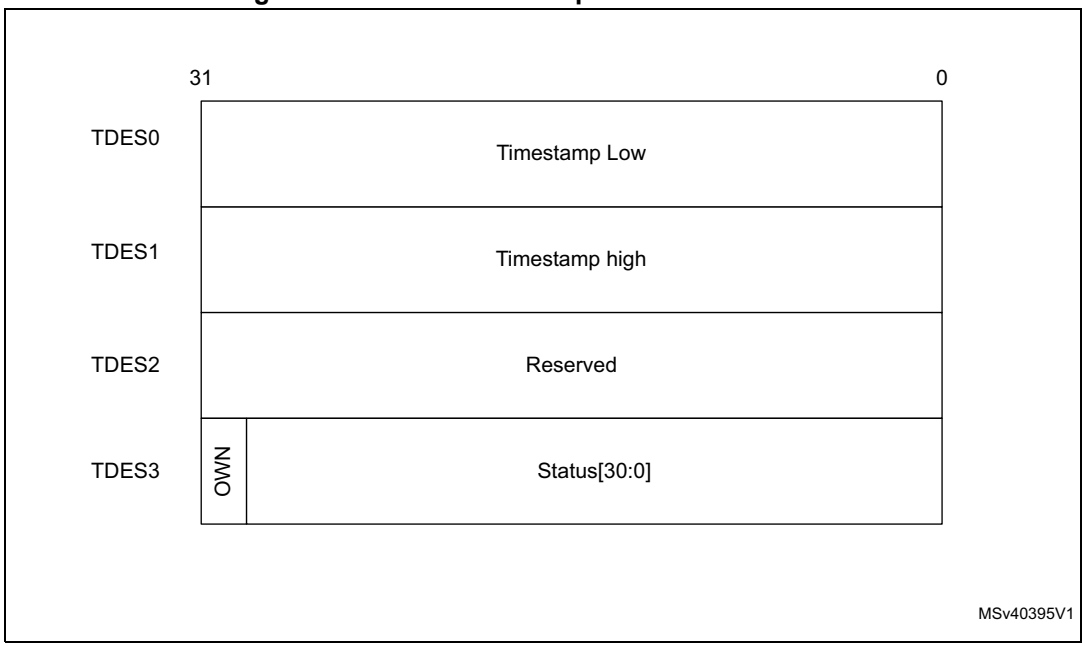
Bits	Name	Description
22:19	SLOTNUM or THL	**SLOTNUM**: Slot Number Control Bits in AV mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field of *Channel x slot function control status register (ETH_DMACxSFCSR)*. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. **THL**: TCP Header Length If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor
18	TSE	**TCP Segmentation Enable** When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	**Checksum Insertion Control or TCP Payload Length** These bits control the checksum calculation and insertion. They can take the following values: 00: Checksum insertion disabled. 01: Only IP header checksum calculation and insertion are enabled. 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.
15	TPL	**Reserved or TCP Payload Length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Reserved or TCP Payload Length** When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/UDP/IP header length. When the TSE bit is reset, this bit is reserved.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 998 shows the write-back format for Transmit normal descriptors. *Table 859* to *Table 862* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 998. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 859. TDES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 860. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 861. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 862. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:24	Reserved	
23	DE	**Descriptor Error** When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor errors can be: – Incorrect sequence from the context descriptor, such as a location after the first descriptor for a packet. – All 1s. – CTXT is set to 1 along with LD or FD bits set to 1. *Note: When a Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel x status register (ETH_DMACxSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.*
22:18	Reserved	

Table 862. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	
15	ES	**Error Summary** This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error
14	JT	**Jabber Timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.

Table 862. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set.
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*. When TBS is enabled in Full duplex mode and this bit is set, it indicates that the frame has been dropped after the expiry time has elapsed.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.

Table 862. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. In Full-duplex mode, when EST/Qbv is enabled and this bit is set, it indicates the frame drop status due to Frame Size error or Schedule Error.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

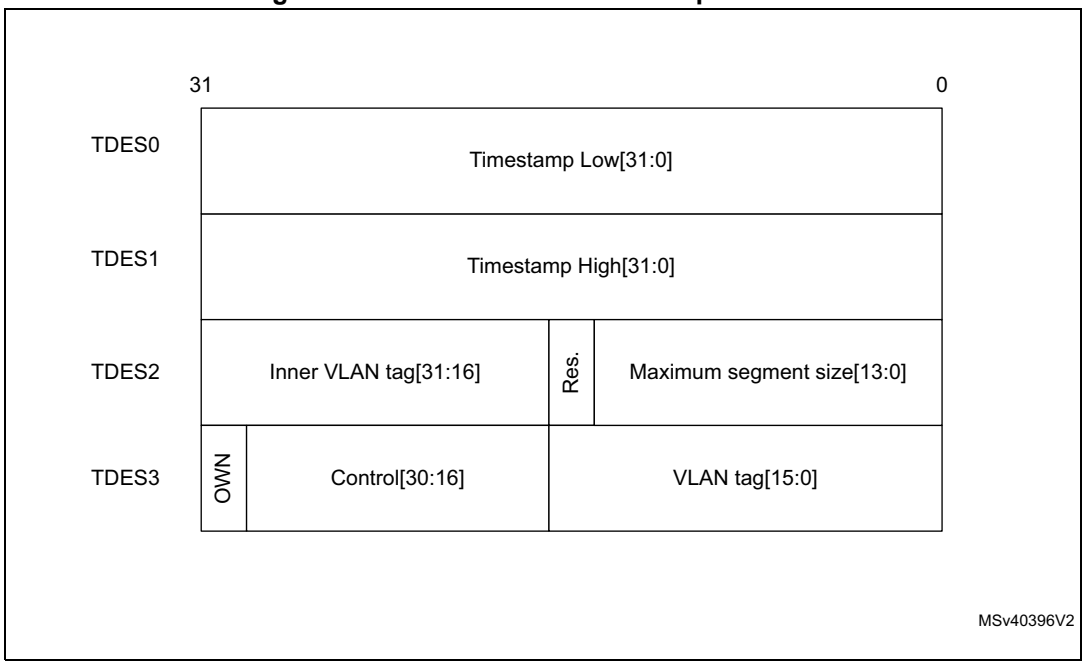
Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 999 shows the format for Transmit context descriptors. *Table 863* to *Table 866* provide a detailed description of all Transmit context descriptors.

Figure 999. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 863. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 864. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 865. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

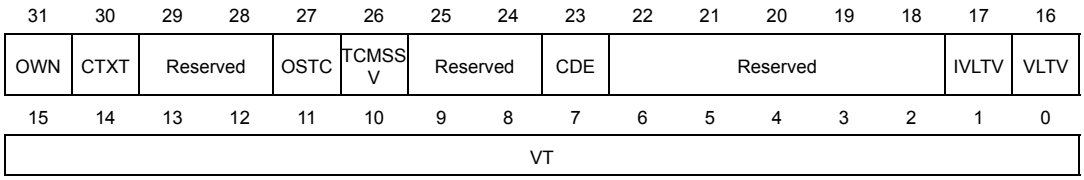


Table 866. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit immediately after a read operation.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	

Table 866. TDES3 context descriptor (continued)

Bit	Name	Description
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor. The Context Descriptor errors can be: – Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. – All 1s. – CD, LD, and FD bits set to 1. *Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel x status register (ETH_DMACxSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.* *This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)*
22:20	Reserved	
19:18	IVTIR	Inner VLAN Tag Insert or Replace When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: 00: Do not add the inner VLAN tag. 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

77.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

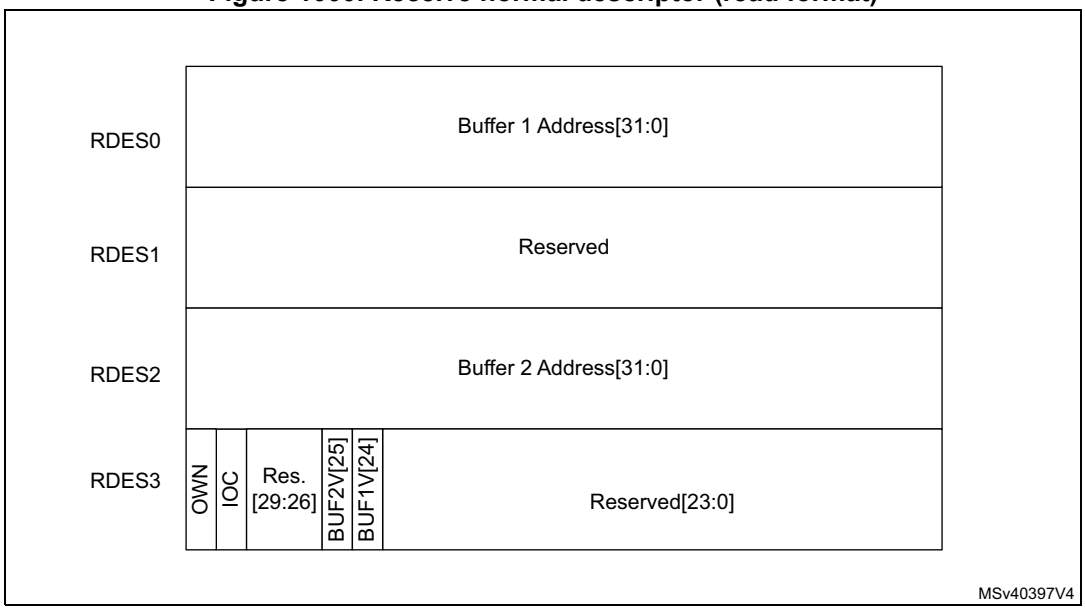
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 1000: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 1001: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 1002: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 1000 shows the read format for Receive normal descriptors. *Table 867* to *Table 870* provide a detailed description of all Receive normal descriptors (read format).

Figure 1000. Receive normal descriptor (read format)



Note: In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 867. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[2:0]=0 in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 868. RDES1 normal descriptor (read format)

Bit	Name	Description
31:0	Reserved or BUF1AP	In 40-bit addressing mode, this field contains the most-significant bits of the Buffer 1 Address Pointer. Otherwise, this field is reserved.

- RDES2 normal descriptor (read format)

Table 869. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[2:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IOC	Reserved				BUF2V	BUF1V	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUF2AP[15:0]															

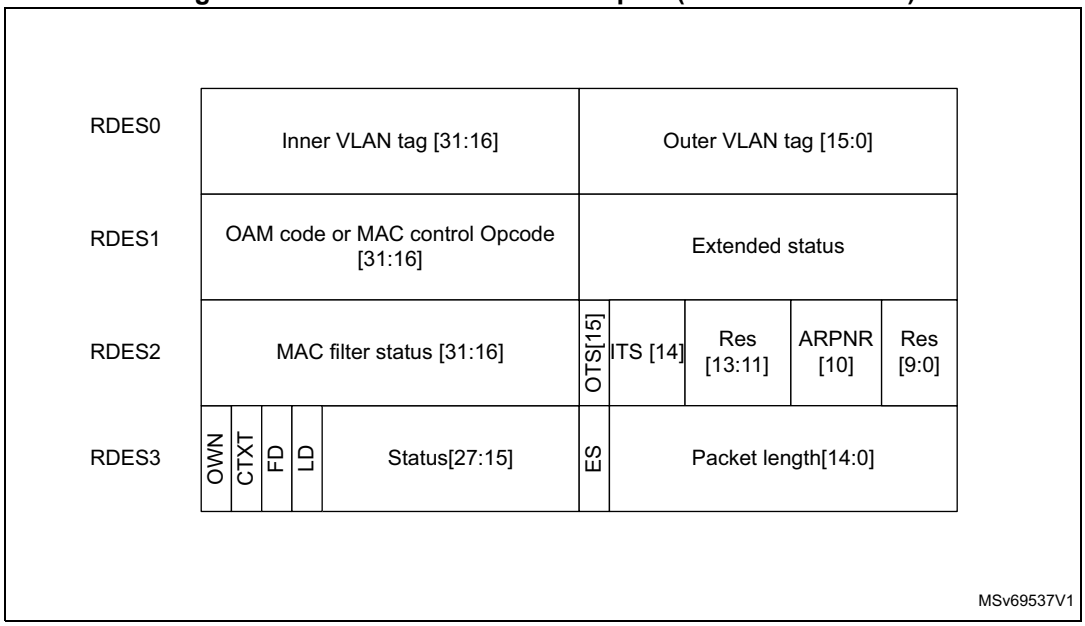
Table 870. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.
23:16	Reserved	
15:0	BUF2AP	**Buffer 2 Address Pointer** In 40-bit addressing mode, this field contains the most-significant bits of the Buffer 2 Address pointer. Otherwise, this field is reserved.

Receive normal descriptor (write-back format)

Figure 1001 shows the write-back format for Receive normal descriptors. *Table 871* to *Table 874* provide a detailed description of all Receive normal descriptors (write-back format).

Figure 1001. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 871. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

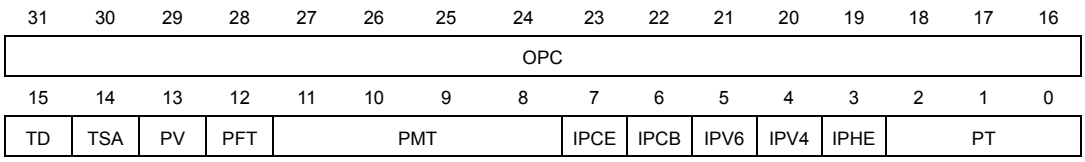


Table 872. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Subtype Code, or MAC Control Packet opcode** OAM Subtype Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Table 872. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** – When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 000: Unknown type or IP/AV payload not processed 001: UDP 010: TCP 011: ICMP 100: IGMP if IPV4 Header Present bit is set 110: AV Tagged Data Packet 111: AV Tagged Control Packet 101: AV Untagged Control Packet Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

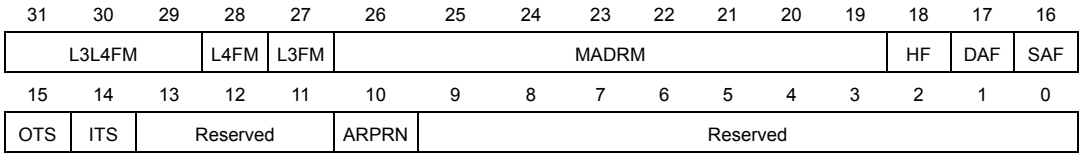


Table 873. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.

Table 873. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	OTS	**Outer VLAN Tag Filter Status** This bit is valid for both Single and Double VLAN tagged frames.
14	ITS	**Inner VLAN Tag Filter Status** This bit is valid only for Double VLAN tagged frames, when Double VLAN Processing is enabled. For more information, see the VLAN Filter Status section.
13:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	PL														

Table 874. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} possible values are: 00: Intermediate Descriptor 01: First Descriptor 10: Reserved 11: Descriptor error (due to all 1s) *Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel x status register (ETH_DMACxSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	**Receive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 874. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
24	CE	**CRC Error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant Packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	**Receive Watchdog Timeout** When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow Error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	**Receive Error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception. This error also includes carrier extension error in the GMII and Half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	**Dribble Bit Error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	**Length/Type Field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000: The packet is a length packet 001: The packet is a type packet. 011: The packet is a ARP Request packet type 100: The packet is a type packet with VLAN Tag 101: The packet is a type packet with Double VLAN tag 110: The packet is a MAC Control packet type 111: The packet is a OAM packet type 010: Reserved

Table 874. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	ES	**Error Summary** When this bit is set, it indicates the logical OR of the following bits: RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[21]: Overflow Error RDES3[22]: Watchdog Timeout RDES3[23]: Giant Packet RDES3[24]: CRC Error This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

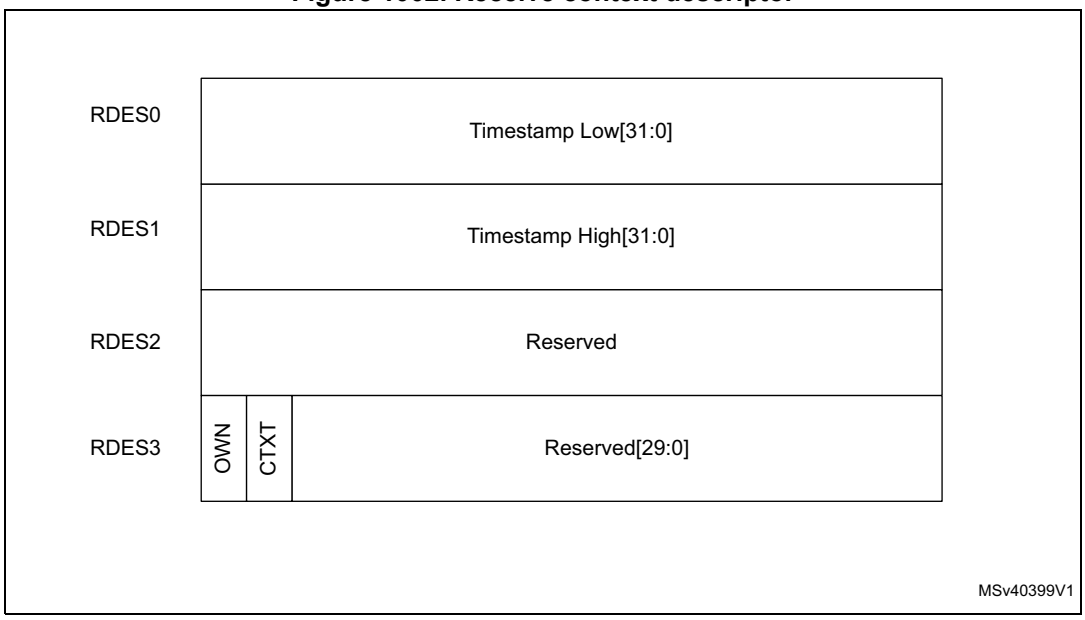
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 1002 shows the format for Receive context descriptors. *Table 875* to *Table 878* provide a detailed description of all Receive context descriptors.

Figure 1002. Receive context descriptor



- RDES0 context descriptor

Table 875. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 876. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 877. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 878. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1: The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor. When CTXT and DE bits are used together, {CTXT, DE} possible values are: 00: Reserved 01: Reserved 10: Context Descriptor 11: Descriptor error When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data The receive DMA sets the CDE bit in *Channel x status register (ETH_DMACxSR)* but does not set the RI field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	DE	**Descriptor Error** Refer to the CTXT bit description for details on how to use the E bit along with CTXT bit.
28:0	Reserved	

77.10.5 Enhanced descriptor for time-based scheduling

The time-based scheduling feature needs 32-byte enhanced descriptors to be enabled on all the DMA channels that intend to use the feature. This is done by setting the EDSE bit of *Channel x transmit control register (ETH_DMACxTXCR)*.

The structure of the 32-byte descriptor for the context and the normal descriptor in read and write formats are described in the following sections.

Enhanced normal descriptor (read format)

Figure 1003. Enhanced normal descriptor (read format)

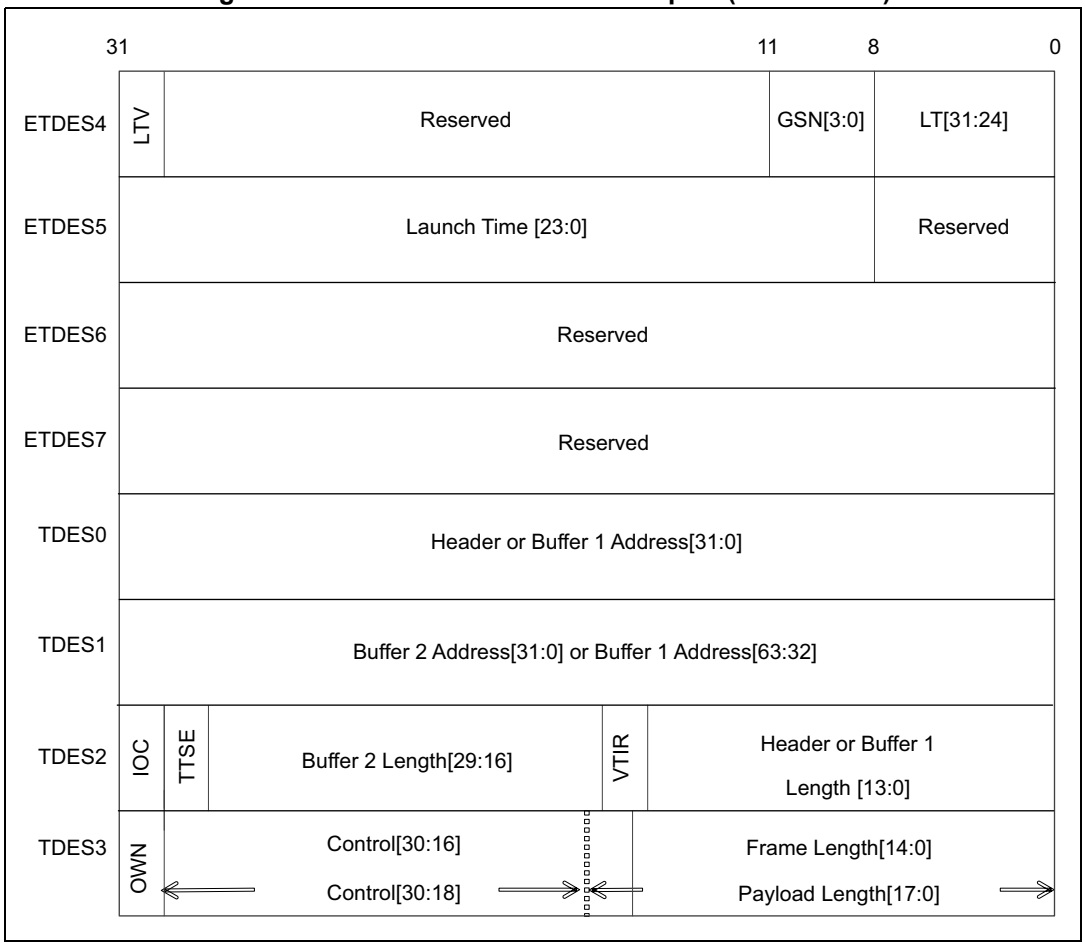


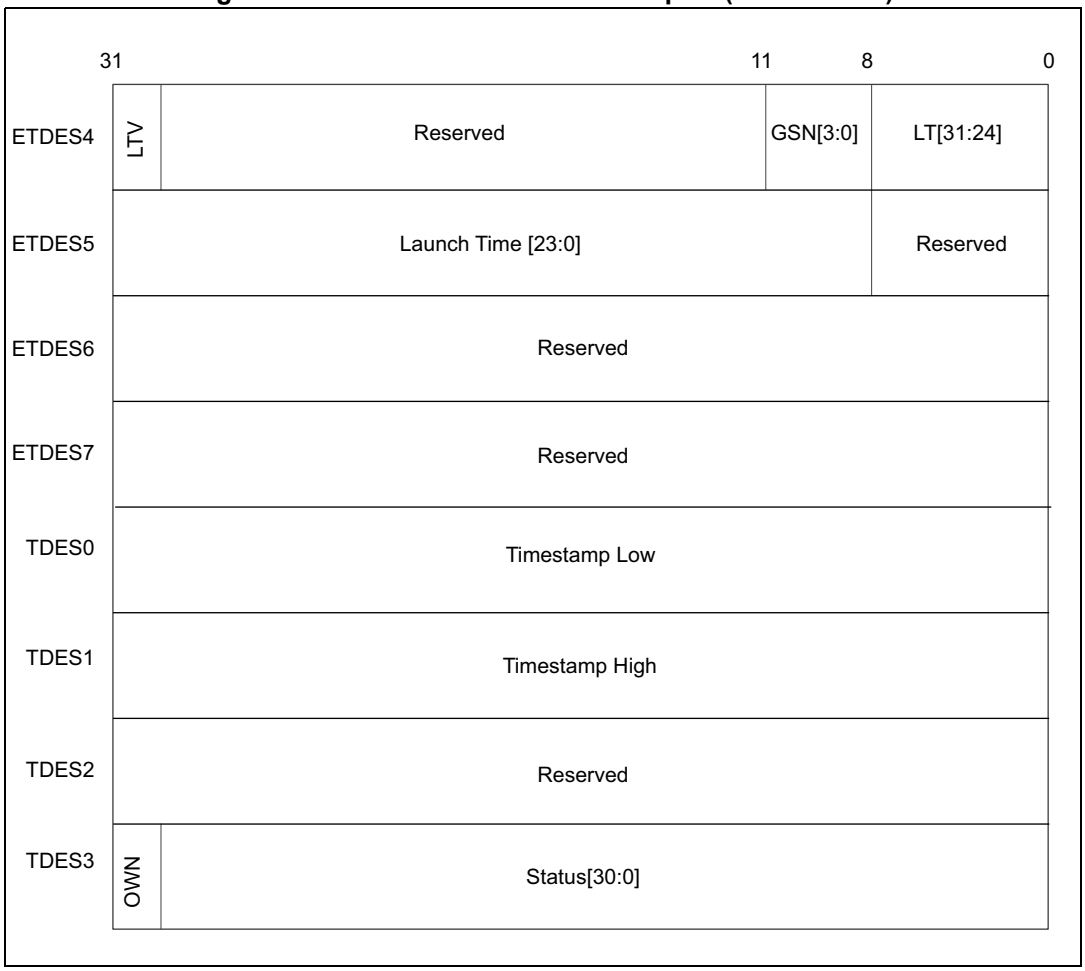
Table 879. Enhanced normal descriptor (read format)

Bit	Name	Description
31	LTV	Launch time valid indicates that the launch time (LT) and GSN fields present in the descriptor are valid. The LTV must be set only if the FD bit of the descriptor is not set.
30:12	Reserved.	
11:8	GSN	GCL slot number associated with the packet
7:0	LT	Launch time associated with the packet.

For details of the other fields, see *Table 855: TDES0 normal descriptor (read format)*.

Enhanced normal descriptor (write format)

Figure 1004. Enhanced normal descriptor (write format)



For details on LTV, GSN, and LT fields, see *Enhanced normal descriptor (read format)*. For details on the other fields, see *Table 859: TDES0 normal descriptor (write-back format)*.

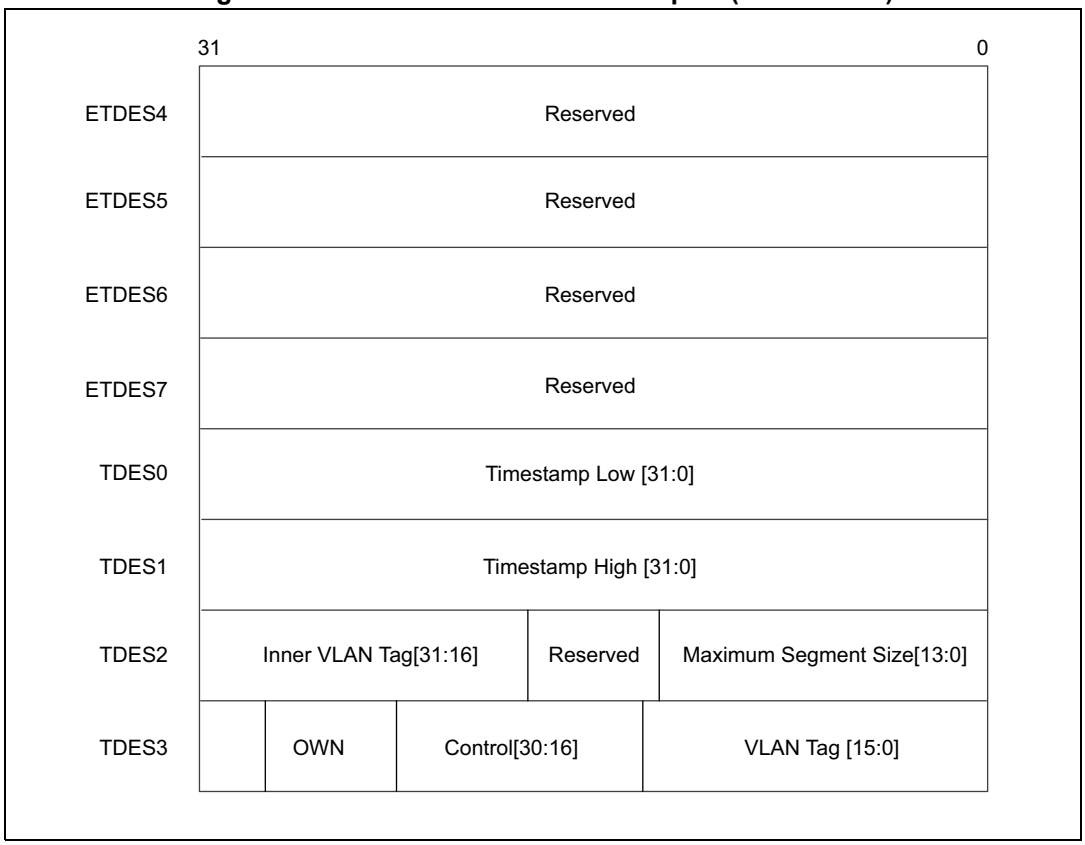
Note: In both write-back formats, no modifications can be done to the extended 16 bytes. The rest of the 16 bytes (TDESC0 to TDESC3) are written back as per the previous 16 bytes descriptor format.

The fetch time, when enabled, overrides the AV slot function.

When an unaligned new GCL list (with a different CTR) is installed, it is recommended not to have traffic during the installation of the new list. Any traffic during the switching of the lists might result in unpredictable behavior of fetch, launch, and launch expiry, as the CTR and BTR values get updated while the frame is being processed.

Enhanced context descriptor (read format)

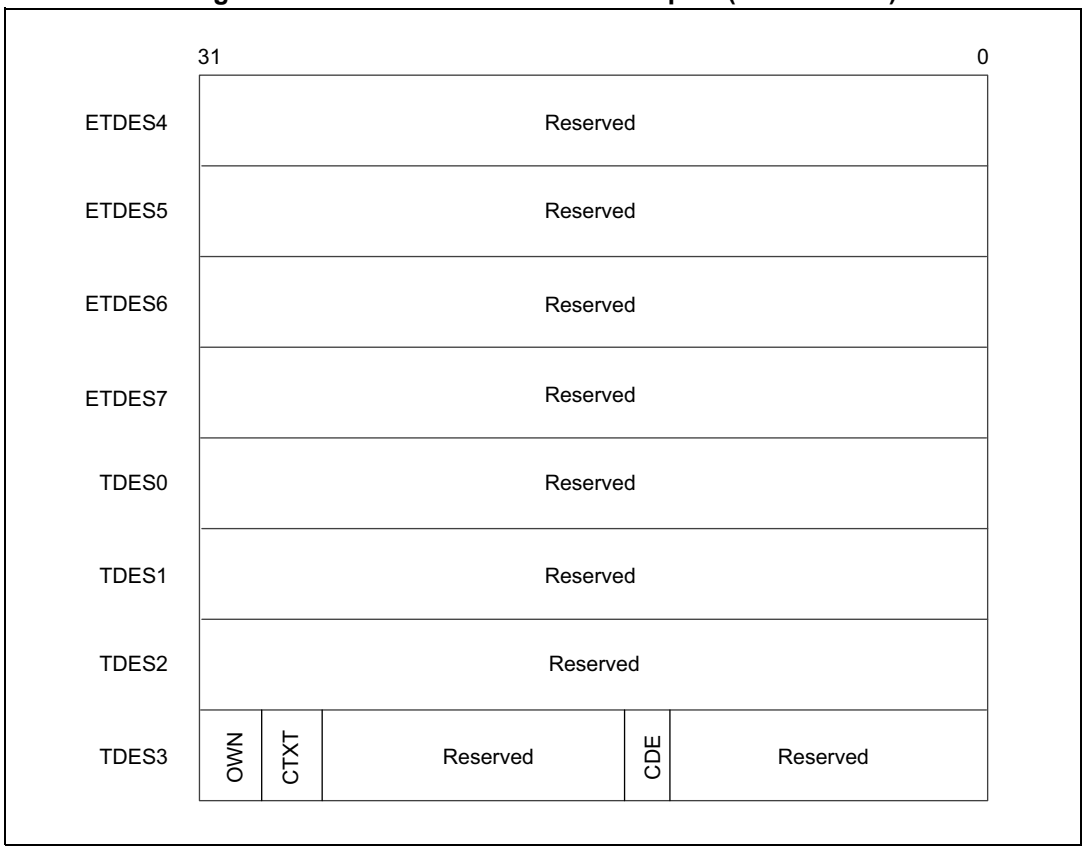
Figure 1005. Enhanced context descriptor (read format)



For details of the various fields, see *Transmit context descriptor*.

Enhanced context descriptor (write format)

Figure 1006. Enhanced context descriptor (write format)



For details of the various fields, see *Receive context descriptor*.

77.11 Ethernet registers

77.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see *Section 77.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 77.11.3: Ethernet MTL registers*)
- MAC registers including MMC registers (see *Section 77.11.4: Ethernet MAC and MMC registers*)

77.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.	TAA2	TAA1	TAA0	Res.	SWR
				rw			rw				r	r	r		rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to *Table 854: Transfer complete interrupt behavior*).

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DSPW**: Descriptor Posted Write

When this bit is set to
0: The descriptor writes are always non-posted.
1: The descriptor writes are non-posted only when IOC (Interrupt on completion) is set in last descriptor. Otherwise, the descriptor writes are always posted.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:2 **TAA[2:0]**: Transmit Arbitration Algorithm

This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected.

000: Fixed priority. In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority.

001: Weighted Strict priority (WSP)

010: Weighted Round-Robin (WRR)

011 to 111: Reserved, must not be used.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0101 0000

The System bus mode register controls the behavior of the AXI master. It mainly controls burst splitting and number of outstanding requests.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]	
rw	rw					rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ONEKBBE	AAL	EAME	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EN_LPI**: Enable Low Power Interface (LPI)

When set to 1, this bit enables the LPI mode and accepts the LPI request from the AXI System Clock controller.

When set to 0, this bit disables the LPI mode and always denies the LPI request from the AXI System Clock controller.

Bit 30 **LPI_XIT_PKT**: Unlock on Magic Packet or Remote wake-up Packet

When set to 1, this bit enables the AXI master to come out of the LPI mode only when the magic packet or remote wake-up packet is received. When set to 0, this bit enables the AXI master to come out of the LPI mode when any packet is received.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **WR_OSR_LMT[1:0]**: AXI Maximum Write Outstanding Request Limit

This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests = WR_OSR_LMT + 1.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **RD_OSR_LMT[1:0]**: AXI Maximum Read Outstanding Request Limit
This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT + 1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ONEKBBE**: 1 Kbyte Boundary Crossing Enable for the AXI Master
When set, the burst transfers performed by the AXI master do not cross 1 Kbyte boundary.
When reset, the burst transfers performed by the AXI master do not cross 4 Kbyte boundary.

Bit 12 **AAL**: Address-Aligned Beats
When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bit 11 **EAME**: Enhanced Address Mode Enable
When set, the DMA master enables the enhanced address mode. In this mode, the DMA engine uses 40-bit addresses.
When cleared, the DMA master enables the normal address mode(32-bit).

Bit 10 **AALE**: Automatic AXI LPI enable
When set to 1, enables the AXI master to enter into LPI state when there is no activity in the Ethernet peripheral for a number of system clock cycles programmed in the LPIEI field of ETH_DMALPIEI register.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **BLEN256**: AXI Burst Length 256
When this bit is set to 1, the AXI master can select a burst length of 256 on the AXI interface.

Bit 6 **BLEN128**: AXI Burst Length 128
When this bit is set to 1, the AXI master can select a burst length of 128 on the AXI interface.

Bit 5 **BLEN64**: AXI Burst Length 64
When this bit is set to 1, the AXI master can select a burst length of 64 on the AXI interface.

Bit 4 **BLEN32**: AXI Burst Length 32
When this bit is set to 1, the AXI master can select a burst length of 32 on the AXI interface.

Bit 3 **BLEN16**: AXI Burst Length 16
When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 16 on the AXI interface.
When the FB bit is set to 0, setting this bit has no effect.

Bit 2 **BLEN8**: AXI Burst Length 8

When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 8 on the AXI interface.

When the FB bit is set to 0, setting this bit has no effect.

Bit 1 **BLEN4**: AXI Burst Length 4

When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 4 on the AXI interface.

When the FB bit is set to 0, setting this bit has no effect.

Bit 0 **FB**: Fixed Burst Length

When this bit is set to 1, the AXI master initiates burst transfers of specified lengths as given below.

– Burst transfers of fixed burst lengths as indicated by the BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, or BLEN4 field

– Burst transfers of length 1: when this bit is set to 0, the AXI master initiates burst transfers that are equal to or less than the maximum allowed burst length programmed in Bits[7:1].

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC3IS	DC2IS	DC1IS	DC0IS
												r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **MACIS**: MAC Interrupt Status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL Interrupt Status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DC3IS**: DMA Channel 3 Interrupt Status

This bit indicates an interrupt event in DMA Channel 3. To reset this bit to 0, the software must read the corresponding register in DMA Channel 3 to get the exact cause of the interrupt and clear its source.

Bit 2 **DC2IS**: DMA Channel 2 Interrupt Status

This bit indicates an interrupt event in DMA Channel 2. To reset this bit to 0, the software must read the corresponding register in DMA Channel 2 to get the exact cause of the interrupt and clear its source.

Bit 1 **DC1IS**: DMA Channel 1 Interrupt Status

This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source.

Bit 0 **DC0IS**: DMA Channel Interrupt Status

This bit indicates an interrupt event in DMA Channel. To reset this bit to 0, the software must read the corresponding register in DMA Channel to get the exact cause of the interrupt and clear its source.

Debug status 1 register (ETH_DMADS1R)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel 0 to 2 for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TPS2[3:0]				Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]			
r	r	r	r					r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRH STS	AXWH STS
r	r	r	r	r	r	r	r							r	r

Bits 31:28 **TPS2[3:0]**: DMA Channel 2 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 2. This field is similar to the TPS0 field.

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **TPS1[3:0]**: DMA Channel 1 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field.

Bits 19:16 **RPS1[3:0]**: DMA Channel 1 Receive Process State
This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field.

Bits 15:12 **TPS0[3:0]**: DMA Channel Transmit Process State
This field indicates the Tx DMA FSM state for Channel:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel Receive Process State
This field indicates the Rx DMA FSM state for Channel:
000: Stopped (Reset or Stop Receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **AXRHSTS**: AXI Master Read Channel Status

When high, this bit indicates that the read channel of the AXI master is active, and it is transferring the data.

Bit 0 **AXWHSTS**: AXI Master Write Channel

When high, this bit indicates that the write channel of the AXI master is active, and it is transferring data.

Debug status 2 register (ETH_DMADS2R)

Address offset: 0x1010

Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel 3 for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS3[3:0]				Res.	Res.	Res.	Res.
								r	r	r	r				

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **TPS3[3:0]**: DMA Channel 3 Transmit Process State

This field indicates the Tx DMA FSM state for Channel 3. This field is similar to the TPS0 field of *Debug status 1 register (ETH_DMADS1R)*.

Bits 3:0 Reserved, must be kept at reset value.

AXI4 transmit channel ACE control register (ETH_DMAA4TXACR)

Address offset: 0x1020

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for read transactions by all the Transmit DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- arcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **THC[3:0]**: Transmit DMA First Packet Buffer or TSO Header Cache Control
When TSO is NOT enabled, This field is used to drive arcache_o[3:0] signal when Transmit DMA is accessing First Buffer of the Packet (First valid buffer with FD being set in the TDES3 of the Descriptor).
When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO Header data.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TEC[3:0]**: Transmit DMA Extended Packet Buffer or TSO Payload Cache Control
When TSO is NOT enabled, This field is used to drive arcache o[3:0] signal when Transmit DMA is accessing the extended buffers (when packet is distributed across multiple buffers).
When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO payload data.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TDRC[3:0]**: Transmit DMA Read Descriptor Cache Control
This field is used to drive arcache_o[3:0] signal when Transmit DMA engines access the Descriptor.

AXI4 receive channel ACE control register (ETH_DMAA4RXACR)

Address offset: 0x1024

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for write transactions by all the Receive DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- awcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **RDC[3:0]**: Receive DMA Buffer Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA is accessing the Buffer when Header and payload are NOT separated.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **RHC[3:0]**: Receive DMA Header Cache Control
This field is used to drive awcache_o[3:0] and signal when Receive DMA is accessing the header Buffer when Header and payload are separated.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RPC[3:0]**: Receive DMA Payload Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA is accessing the Payload Buffer when Header and payload are separated.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **RDWC[3:0]**: Receive DMA Write Descriptor Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA accesses the Descriptor.

AXI4 descriptor ACE control register (ETH_DMAA4DACR)

Address offset: 0x1028

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for Descriptor write transactions by all the TxDMA channels and Descriptor read transactions by all the RxDMA channels. It also controls the values to be driven on awprot_m_o and arprot_m_o.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **RDRC[3:0]**: Receive DMA Read Descriptor Cache control
This field is used to drive arcache_o[3:0] signal when Receive DMA engines read the Descriptor.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TDWD[1:0]**: Transmit DMA Write Descriptor Domain control
This field is used to drive awdomain_o[1:0] signal when Transmit DMA write to the Descriptor.

Bits 3:0 **TDWC[3:0]**: Transmit DMA Write Descriptor Cache control
This field is used to drive awcache_o[3:0] signal when Transmit DMA writes to the Descriptor.

AXI4 LPI Entry Interval register (ETH_DMALPIEI)

Address offset: 0x1040

Reset value: 0x0000 0000

This register is used to control the AXI LPI entry interval.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **LPIEI[3:0]**: LPI Entry Interval

Contains the number of system clock cycles, multiplied by 64, to wait for an activity in the peripheral to enter into the AXI low power state.
0 indicates 64 clock cycles.

DMA TBS control register 0 (ETH_DMATBSCTRL0R)

Address offset: 0x1050

Reset value: 0x0000 0000

This register controls time-based scheduling attributes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTOS[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTOS[7:0]								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				rw

Bits 31:8 **FTOS[23:0]**: Fetch time offset

The value in units of 256 nanoseconds, that has to be deducted from the launch time to compute the fetch time.
The maximum value is 999,999,999 ns. It must be smaller than CTR - 1 when the ESTM mode is set since this value is a modulo CTR value.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **FGOS[2:0]**: Fetch GSN offset

The number of GSN slots that must be deducted from the launch GSN to compute the fetch GSN. This value is valid only when FTOV is set.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **FTOV**: Fetch time offset valid

When set, this bit indicates that the FTOS field is valid. When not set, it indicates that the fetch offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.

0: Fetch time offset invalid

1: Fetch time offset valid

DMA TBS control register x (ETH_DMATBSCTRLxR)

Address offset: 0x1050 + 0x04 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register controls time-based scheduling attributes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTOS[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTOS[7:0]								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				rw

Bits 31:8 **FTOS[23:0]**: Fetch time offset

The value in units of 256 nanoseconds, that has to be deducted from the launch time to compute the fetch time.

The maximum value is 999,999,999 ns. It must be smaller than CTR - 1 when the ESTM mode is set since this value is a modulo CTR value.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **FGOS[2:0]**: Fetch GSN offset

The number of GSN slots that must be deducted from the launch GSN to compute the fetch GSN. This value is valid only when FTOV is set.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **FTOV**: Fetch time offset valid

When set, this bit indicates that the FTOS field is valid. When not set, it indicates that the fetch offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.

0: Fetch time offset invalid

1: Fetch time offset valid

Channel x control register (ETH_DMACxCR)

Address offset: 0x1100+ 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The DMA Channel x control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8
											rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MSS[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 64-bit double-word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in *Channel x transmit control register (ETH_DMACxTXCR)* and in Bits[21:16] in *Channel x receive control register (ETH_DMACxRXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel x transmit control register (ETH_DMACxTXCR)* is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

Channel x transmit control register (ETH_DMACxTXCR)

Address offset: 0x1104 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The DMA Channel x Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFSEL 1	TFSEL 0	EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
rw			rw								rw	r	r	r	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **TFSEL[1:0]**: TBS Fetch Select

This field selects one of the four ETH_DMATBSCTRLxR register fields (FTOS,FGSN,FTOV) for the channel.

00: ETH_DMATBSCTRL0R

01: ETH_DMATBSCTRL1R

10: ETH_DMATBSCTRL2R

11: ETH_DMATBSCTRL3R

Bit 28 **EDSE**: Enhanced Descriptor Enable

When this bit is set, the corresponding channel uses 32-byte enhanced descriptors both for normal and context descriptors.

When reset, the corresponding channel uses the 16-byte descriptors.

Bits 27:24 **TQOS[3:0]**: Transmit QOS

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Tx Channel.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in ETH_DMACxCR.

b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 8192 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 **IPBL**: Ignore PBL Requirement

When this bit is set, the DMA does not check for PBL number of locations in the MTL before initiating a transfer. If space is not available, the MTL might use handshaking to slow the DMA.

Note: This bit/mode must not be used when multiple Transmit DMA Channels are enabled as it might block other Transmit and Receive DMA Channels from accessing the Read Data Channel of AXI bus until space is available in Transmit Queue for current transfer.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in Channel. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 **TCW[2:0]**: Transmit Channel Weight

This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the base address of the Transmit list set by the ETH_DMACxTXDLAR register.
- The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACxSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACxTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACxTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel x receive control register (ETH_DMACxRXCR)

Address offset: 0x1108 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA Channel x Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	Res.	Res.	Res.	RQOS[3:0]				Res.	Res.	RXPBL[5:0]					
rw				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RBSZ[13:0]														SR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RPF**: DMA Rx Channel x Packet Flush

When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx Channel x when the DMA Rx Channel x is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx queue.

When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx Channel x after the DMA is stopped due to a system bus error.

This might cause head-of-line blocking in the corresponding RxQueue.

Note: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in ETH_DMACxRXCR register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **RQOS[3:0]**: Rx AXI4 QOS.

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Rx Channel.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMACCR.

b) Set the RXPBL[5:0].

Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue x operating mode register (ETH_MTLRXQxOMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queue Controller is transferring data to MAC.The total locations in Rx Queue of size 4096 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

Note: The buffer size must be a multiple of 8. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 8, it may result into undefined behavior.

The LSB bits (2:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMACSR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel x Tx descriptor list high address register (ETH_DMACxTXDLHAR)

Address offset: 0x1110 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Channel x Tx Descriptor List High Address register has the higher bits of the start address of Transmit descriptor list.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in the ETH_DMACxTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier. This register must be programmed with the desired higher address before programming the *Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDESHA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDESHA[7:0]**: Start of Transmit List

This bitfield contains the most-significant eight bits of the 40-bit base address of the first descriptor in the Transmit descriptor list.

Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)

Address offset: 0x1114+ 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Channel x Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMACxTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDESLA[31:0]**: Start of Transmit List

This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel x Rx descriptor list high address register (ETH_DMACxRXDLHAR)

Address offset: 0x1118 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Rx Descriptor List Address register has the higher bits of the start address of Receive descriptor list.

Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, the SR bit is cleared in ETH_DMACxRXCR register.

When the SR bit is set, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDESHA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDESHA[7:0]**: Start of Receive List

This field contains the most-significant eight bits of the 40-bit base address of the first descriptor in the Rx Descriptor list.

Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)

Address offset: 0x111C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACxRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDESLA[31:0]**: Start of Receive List

This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)

Address offset: 0x1120 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Channel x Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDT[31:0]**: Transmit Descriptor Tail Pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)

Address offset: 0x1128 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)

Address offset: 0x112C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TDRL[9:0]**: Transmit Descriptor Ring Length

This 10-bit field sets the maximum number of Tx descriptors in the circular descriptor ring.
TDRL[9:0] corresponds to the number of Tx descriptors minus one.
It can be programmed to any value up to 0x3FF (1024 descriptors).
A minimum ring descriptor length of four is recommended.
For example, program 0x9 to configure 10 descriptors.

Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)

Address offset: 0x1130 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ARBS[6:0]**: Alternate Receive Buffer Size

Indicates size in bytes for Buffer 1 when ARBS[6:0] is programmed to a non-zero value.
When ARBS[6:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of *Channel x receive control register (ETH_DMACxRXCR)*.

Bits 16:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This 10-bit field sets the maximum number of Rx descriptors in the circular descriptor ring.
RDRL[9:0] corresponds to the number of Rx descriptors minus one. It can be programmed to any value up to 0x3FF (1024 descriptors).
For example, program 0x9 to configure 10 descriptors.

Channel x interrupt enable register (ETH_DMACxIER)

Address offset: 0x1134 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Channel x Interrupt Enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt (not available for Channels 2 and 3)
Bit 11: Early Receive interrupt (not available for Channels 2 and 3)
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:
Bit 1: Transmit Process Stopped
Bit 7: Rx Buffer Unavailable (not available for Channels 2 and 3)
Bit 8: Receive Process Stopped (not available for Channels 2 and 3)
Bit 9: Receive Watchdog Timeout (not available for Channels 2 and 3)
Bit 10: Early transmit interrupt
Bit 12: Fatal Bus Error
Bit 13: Context Descriptor Error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context Descriptor Error Enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error Enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early receive interrupt Enable

When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled. This bit is not available for Channels 2 and 3.

Bit 10 **ETIE**: Early Transmit Interrupt Enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive Watchdog Timeout Enable

When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled. This bit is not available for Channels 2 and 3.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled. This bit is not available for Channels 2 and 3.

Bit 7 **RBUE**: Receive Buffer Unavailable Enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled. This bit is not available for Channels 2 and 3.

Bit 6 **RIE**: Receive Interrupt Enable

When this bit is set along with the NIE bit, the receive interrupt is enabled. When this bit is reset, the receive interrupt is disabled. This bit is not available for Channels 2 and 3.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable Enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

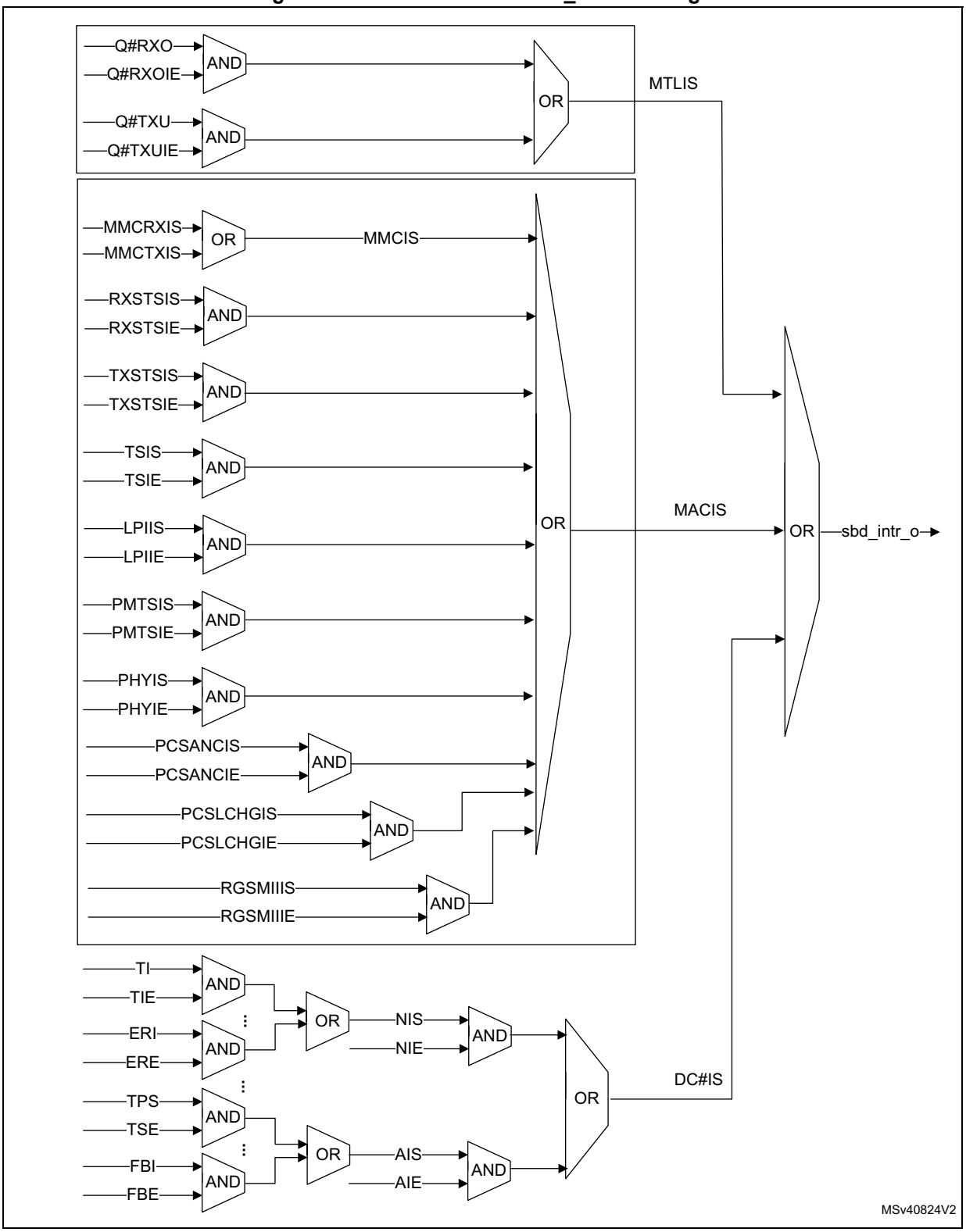
Bit 1 **TXSE**: Transmit Stopped Enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt Enable

When this bit is set along with the NIE bit, the transmit interrupt is enabled. When this bit is reset, the transmit interrupt is disabled.

Figure 1007. Generation of ETH_DMAISR flags



Channel x Rx interrupt watchdog timer register (ETH_DMACxRXIWTR)

Address offset: 0x1138 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The receive interrupt Watchdog Timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the *Channel x status register (ETH_DMACxSR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in RWT[7:0] bitfield.
00: 256
01: 512
10: 1024
11: 2048
For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This bitfield indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.
The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACxSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].
When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel x slot function control status register (ETH_DMACxSFCSR)

Address offset: 0x113C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 07C0

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIV[11:0]												Res.	Res.	ASC	ESC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **RSN[3:0]**: Reference Slot Number

This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.

Bits 15:4 **SIV[11:0]**: Slot Interval Value

This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 µs while the maximum value 4095 specifies the slot interval of 4096 µs. The default/reset value is 0x07C which corresponds to slot interval of 125 µs.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ASC**: Advance Slot Check

When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is

– equal to the reference slot number given in the RSN field

or

– ahead of the reference slot number by up to two slots

This bit is applicable only when the ESC bit is set.

Bit 0 **ESC**: Enable Slot Comparison

When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is

– equal to the reference slot number

or

– ahead of the reference slot number by one slot

When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.

Note: The UFO (UDP Fragmentation over IPv4)/TSO/USO should not be enabled along with TBS/AVB Slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application and the slot number check are applicable for fetching of only first segment/fragment. As a result it might be difficult for the software to specify the slot number for subsequent packets.

Channel x current application transmit descriptor register (ETH_DMACxCATXDR)

Address offset: 0x1144 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Channel x Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTDESAPTR[31:0]**: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel x current application receive descriptor register (ETH_DMACxCARXDR)

Address offset: 0x114C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRDESAPTR[31:0]**: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x current application transmit buffer high register (ETH_DMACxCATXBHR)

Address offset: 0x1150 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Channel x Current Application Transmit Buffer Address High register has the higher bits of the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURTBUFAPTRH[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CURTBUFAPTRH[7:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel x current application transmit buffer register (ETH_DMACxCATXBR)

Address offset: 0x1154 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Channel x Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel x current application receive buffer high register (ETH_DMACxCARXBHR)

Address offset: 0x1158 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Current Application Receive Buffer Address High register has the higher bits of the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURRBUFAPTRH[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CURRBUFAPTRH[7:0]**: Application Receive Buffer Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x current application receive buffer register (ETH_DMACxCARXBR)

Address offset: 0x115C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRBUFAPTR[31:0]**: Application Receive Buffer Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x status register (ETH_DMACxSR)

Address offset: 0x1160 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rw	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.

Bit [2]: Error during data transfer by Rx DMA when 1, no rror during data transfer by Rx DMA when 0.

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt. This bit is not available for Channels 2 and 3.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.

Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, Error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:

Bit 0: Transmit interrupt

Bit 2: Transmit buffer unavailable

Bit 6: Receive interrupt (not available for Channels 2 and 3)

Bit 11: Early receive interrupt (not available for Channels 2 and 3)

Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACxIER register) affect the Normal Interrupt Summary bit.

This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:

Bit 1: Transmit Process Stopped

Bit 7: Receive Buffer Unavailable (not available for Channels 2 and 3)

Bit 8: Receive Process Stopped (not available for Channels 2 and 3)

Bit 10: Early transmit interrupt

Bit 12: Fatal Bus Error

Bit 13: Context Descriptor Error

Only unmasked bits affect the Abnormal Interrupt Summary bit.

This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context Descriptor Error

This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one‘s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12 **FBE**: Fatal Bus Error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early Receive Interrupt

This bit when set indicates that the RxDMA has completed the transfer of packet data to the memory. The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early Transmit Interrupt

This bit when set indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory.

Bit 9 **RWT**: Receive Watchdog Timeout

This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received. This bit is not available for Channels 2 and 3.

Bit 8 **RPS**: Receive Process Stopped

This bit is asserted when the Rx process enters the Stopped state. This bit is not available for Channels 2 and 3.

Bit 7 **RBU**: Receive Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. This bit is not available for Channels 2 and 3.

Bit 6 **RI**: Receive Interrupt

This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

The reception remains in the Running state. This bit is not available for Channels 2 and 3.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the *Debug status 1 register (ETH_DMADS1R)* register explains the Transmit Process state transitions.

To resume processing the Transmit descriptors, the application should do the following:

1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.

For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.

Bit 1 **TPS**: Transmit Process Stopped

This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit Interrupt

This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel x missed frame count register (ETH_DMACxMFCR)

Address offset: 0x1164 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in *Channel x receive control register (ETH_DMACxRXCR)* register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
rc_r					rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped Packet Counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in *Channel x receive control register (ETH_DMACxRXCR)*. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 880. ETH_DMA common register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]		Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.				Res.	SWR
	Reset value															0	0					0			0								0
0x1004	ETH_DMASBMR	EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]		Res.	Res.	ONEKBBE	AAL	EAME	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
	Reset value	0	0					0	1							0	1			0	0	0	0			0	0	0	0	0	0	0	0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC3IS	DC2IS	DC1IS	DC0IS
	Reset value															0	0													0	0	0	0
0x100C	ETH_DMADS1R	TPS2[3:0]				Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]				TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRHSTS	AXWHSTS
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0
0x1010	ETH_DMADS2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS3[3:0]				Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x1014 - 0x101C	Reserved																																
0x1020	ETH_ DMAA4TXACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]				Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
	Reset value													0	0	0	0					0	0	0	0					0	0	0	0
0x1024	ETH_ DMAA4RXACR	Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]				Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x1028	ETH_ DMAA4DACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
	Reset value																					0	0	0	0			0	0	0	0	0	0
0x102C - 0x103C	Reserved																																
0x1040	ETH_DMALPIEI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
	Reset value																													0	0	0	0

Table 880. ETH_DMA common register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1044 - 0x104C	Reserved																																
0x1050	ETH_DMATBSCTRL0R	FTOS[23:0]																								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0				0
0x1050 + 0x4 * (x = 0 to 3) Last 0x105C	ETH_DMATBSCTRLxR	FTOS[23:0]																								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0				0

Table 881. ETH_DMA_CH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMAC0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_DMAC0TXCR	Res.	TFSEL[1:0]		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0			0								0	0	0	0	0
0x1108	ETH_DMACRXCR	RPF	Res.	Res.	Res.	RQOS[3:0				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]												Res.	Res.	Res.	SR
	Reset value	0				0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0				0
0x110C	Reserved																																
0x1110	ETH_DMAC0TXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1114	ETH_DMAC0TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1118	ETH_DMAC0RXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x111C	ETH_DMAC0RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1120	ETH_DMAC0TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1124	Reserved																																

Table 881. ETH_DMA_CH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1128	ETH_DMAC0RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x112C	ETH_DMAC0TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_DMAC0RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMAC0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_DMACRXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x113C	ETH_DMAC0SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0			0	0
0x1140	Reserved																																
0x1144	ETH_DMAC0CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x0114C	ETH_DMACCARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	ETH_DMAC0CATXBHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURTBUFAPTRH[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1154	ETH_DMAC0CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	ETH_DMAC0CARXBHR	CURRBUFAPTRH[7:0]																															
	Reset value																									0	0	0	0	0	0	0	0
0x115C	ETH_DMAC0CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMAC0SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1164	ETH_DMAC0MFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Table 882. ETH_DMA_CH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1180	ETH_DMAC1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1184	ETH_DMAC1TXCR	Res.	TFSEL[1:0]		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0			0								0	0	0	0	0
0x1188	ETH_ DMAC1RXCR	RPF	Res.	Res.	Res.	RQOS[3:0				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]											Res.	Res.	Res.	SR
	Reset value	0				0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x118C	Reserved																																
0x1190	ETH_ DMAC1TXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1194	ETH_ DMAC1TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1198	ETH_ DMAC1RXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x119C	ETH_ DMAC1RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11A0	ETH_DMAC1TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11A4	Reserved																																
0x11A8	ETH_ DMAC1RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11AC	ETH_ DMAC1TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x11B0	ETH_ DMAC1RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0
0x11B4	ETH_ DMAC1IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11B8	ETH_ DMAC0RXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0

Table 882. ETH_DMA_CH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x11BC	ETH_DMAC1SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0
0x11C0	Reserved																																
0x11C4	ETH_DMAC1CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C8	Reserved																																
0x11CC	ETH_DMACCARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11D0	ETH_DMAC1CATXBHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURTBUFAPTRH[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x11D4	ETH_DMAC1CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11D8	ETH_DMAC1CARXBHR	CURRBUFAPTRH[7:0]																															
	Reset value																									0	0	0	0	0	0	0	0
0x11DC	ETH_DMAC1CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11E0	ETH_DMAC1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11E4	ETH_DMAC1MFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Table 883. ETH_DMA_CH2 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1200	ETH_DMAC2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1204	ETH_DMAC2TXCR	Res.	TFSEL[1:0]		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]		ST
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0			0									0	0	0	0	0
0x1208	Reserved																																
0x1210	ETH_DMAC2TXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 883. ETH_DMA_CH2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1214	ETH_DMAC2TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1218 - 0x1220	Reserved																																
0x1224	ETH_DMAC2TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1228	Reserved																																
0x122C	ETH_DMAC2TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1230	Reserved																																
0x1234	ETH_DMAC2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1238	Reserved																																
0x123C	ETH_DMAC2SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0
0x1240	Reserved																																
0x1244	ETH_DMAC2CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1248	Reserved																																
0x1250	ETH_DMAC2CATXBHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURTBUFAPTRH[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1254	ETH_DMAC2CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1258 - 0x125C	Reserved																																
0x1260	ETH_DMAC2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0

Table 884. ETH_DMA_CH3 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1280	ETH_DMAC3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x1284	ETH_DMAC3TXCR	Res.	TFSEL[1:0]		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]					IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST	
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0			0							0	0	0	0	0
0x1288	Reserved																																
0x1290	ETH_ DMAC3TXDLHAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDESHA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1294	ETH_ DMAC3TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1298 - 0x12A0	Reserved																																
0x12A4	ETH_DMAC3TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12A8	Reserved																																
0x12AC	ETH_ DMAC3TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x12B0	Reserved																																
0x12B4	ETH_ DMAC3IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x12B8	Reserved																																
0x12BC	ETH_ DMAC3SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0
0x12C0	Reserved																																
0x12C4	ETH_ DMAC3CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C8	Reserved																																
0x12D0	ETH_ DMAC3CATXBHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURTBUFAPTRH[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x12D4	ETH_ DMAC3CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12D8 - 0x12DC	Reserved																																

Table 884. ETH_DMA_CH3 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x12E0	ETH_DMAC3SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

77.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CNT CLR	CNT PRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTX STS	Res.
						rw	rw		rw	rw			rw	rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters Preset

When this bit is set:

- *Tx queue x underflow register (ETH_MTLTXQxUR)* is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in *Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)* is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **SCHALG[1:0]**: Tx Scheduling Algorithm

This field indicates the algorithm for Tx scheduling:
00: Weighted round robin (WRR) algorithm
01 to 10: Reserved, must not be used.
11: Strict priority (SP) algorithm.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RAA**: Receive Arbitration Algorithm

This field is used to select the arbitration algorithm for the Rx side.
0: Strict priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority.
1: Weighted Strict priority (WSP)

Bit 1 **DTXSTS**: Drop Transmit Status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.

Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESTIS	Res.	Res.
													r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q3IS	Q2IS	Q1IS	Q0IS
												r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ESTIS**: EST (TAS- 802.1Qbv) Interrupt Status
This bit indicates an interrupt event during the operation of 802.1Qbv. To reset this bit, the application must clear the error/event that has caused the Interrupt.

Bits 17:4 Reserved, must be kept at reset value.

Bit 3 **Q3IS**: Queue 3 interrupt status
This bit indicates that an interrupt has been generated by Queue 3. To reset this bit, read ETH_MTLQ1ICSR register to identify the interrupt cause and clear the source.

Bit 2 **Q2IS**: Queue 2 interrupt status
This bit indicates that an interrupt has been generated by Queue 2. To reset this bit, read ETH_MTLQ1ICSR register to identify the interrupt cause and clear the source.

Bit 1 **Q1IS**: Queue 1 interrupt status
This bit indicates that an interrupt has been generated by Queue 1. To reset this bit, read ETH_MTLQ1ICSR register to identify the interrupt cause and clear the source.

Bit 0 **Q0IS**: Queue 0 interrupt status
This bit indicates that an interrupt has been generated by Queue 0. To reset this bit, read ETH_MTLQ0ICSR register to identify the interrupt cause and clear the source.

Rx Queue and DMA Channel Mapping register (ETH_MTLRXQDMAMR)

Address offset: 0x0C30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Q1DDMACH	Res.	Res.	Res.	Q1MDMACH	Res.	Res.	Res.	Q0DDMACH	Res.	Res.	Res.	Q0MDMACH
			rw				rw				rw				rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **Q1DDMACH**: Queue 1 Enabled for DA-based DMA Channel Selection

When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.
When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programmed in the Q1MDMACH bit.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **Q1MDMACH**: Queue 1 Mapped to DMA Channel

This field controls the routing of the received packet in Queue 1 to the DMA channel:
0: DMA Channel 0
1: DMA Channel 1
This field is valid when the Q1DDMACH field is reset.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **Q0DDMACH**: Queue 0 Enabled for DA-based DMA Channel Selection

When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.
When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programmed in the Q0MDMACH bit.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **Q0MDMACH**: Queue 0 Mapped to DMA Channel

This field controls the routing of the received packet in Queue 0 to the DMA channel:
0: DMA Channel 0
1: DMA Channel 1
This field is valid when the Q0DDMACH field is reset.

TBS control register (ETH_MTLTBSCR)

Address offset: 0x0C40

Reset value: 0x0000 0000

This register controls the time-based scheduling operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LEOS[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LEOS[7:0]								Res.	LEGOS[2:0]			Res.	Res.	LEOV	ESTM
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:8 **LEOS[23:0]**: Launch Expiry Offset

The value in units of 256 nanoseconds that has to be added to the launch time to compute the launch expiry time. The value is valid only when LEOV is set.

The maximum value is 999,999,999 ns. It must be smaller than CTR - 1 when the ESTM mode is set since this value is a modulo CTR value.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **LEGOS[2:0]**: Launch Expiry GSN Offset

The number of GSN slots that has to be added to the launch GSN to compute the launch expiry time. This value is valid only when LEOV is set.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **LEOV**: Launch expiry offset valid

When set, this bit indicates that the LEOS field is valid. When not set, it indicates that the launch expiry offset is not valid and that the MTL must not check for launch expiry time.

0: LEOS field invalid

1: LEOS field valid

Bit 0 **ESTM**: EST offset mode

When this bit is set, the launch time value used in time-based scheduling is interpreted as an EST offset value and is added to the base time register (BTR) of the current list.

When reset, the launch time value is used as an absolute value that must be compared with the system time [39:8].

0: EST offset mode disabled

1: EST offset mode enabled

EST Control register (ETH_MTLESTCR)

Address offset: 0x0C50

Reset value: 0x0000 0000

This register controls the operation of Enhancements to Scheduled Transmission (IEEE802.1Qbv).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTOV[7:0]								CTOV[11:4]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTOV[3:0]				Res.	TILS[2:0]			LCSE[1:0]		DFBS	DDBF	Res.	Res.	SSWL	EEST
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:24 **PTOV[7:0]**: PTP Time Offset Value

The value of PTP Clock period multiplied by 6 in nanoseconds. This value is needed to avoid transmission overruns at the beginning of the installation of a new GCL.

Bits 23:12 **CTOV[11:0]**: Current Time Offset Value

Provides a 12 bit time offset value in nanosecond that is added to the current time to compensate for all the implementation pipeline delays such as the CDC sync delay, buffering delays, data path delays and so on. This offset helps to ensure that the impact of gate controls is visible on the line exactly at the predetermined schedule (or as close to the schedule as possible).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **TILS[2:0]**: Time Interval Left Shift Amount

This field provides the left shift amount for the programmed Time Interval values used in the Gate Control Lists.
000: No left shift needed (equal to x1ns)
001: Left shift TI by 1 bit (equal to x2ns)
010: Left shift TI by 2 bits (equal to x4ns)
...
100: Left shift TI by 7 bits (equal to x128ns)

Bits 7:6 **LCSE[1:0]**: Loop Count to report Scheduling Error

Programmable number of GCL list iterations before reporting an HLBS error defined in *EST Status register (ETH_MTLESTSR)*.
00: 4 iterations
01: 8 iterations
10: 16 iterations
11: 32 iterations

Bit 5 **DFBS**: Drop Frames causing Scheduling Error

When set frames reported to cause HOL Blocking due to not getting scheduled (HLBS field of *EST Status register (ETH_MTLESTSR)*) after 4,8,16,32 (based on LCSE field of this register) GCL iterations are dropped.
0: Do not drop
1: Drop

Bit 4 **DDBF**: Do not Drop frames during Frame Size Error

When set, frames are not be dropped during Head-of-Line blocking due to Frame Size Error (HLBF field of *EST Status register (ETH_MTLESTSR)*).

0: Drop

1: Do not drop

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SSWL**: Switch to S/W owned list

When set, indicates that the software has programmed that list that it currently owns (SWOL) and the hardware should switch to the new list based on the new BTR. Hardware clears this bit when the switch to the SWOL happens to indicate the completion of the switch or when an BTR error (BTRE in *EST Status register (ETH_MTLESTSR)*) is set. When BTRE is set this bit is cleared but SWOL is not updated as the switch was not successful.

Bit 0 **EEST**: Enable EST

When reset, the gate control list processing is halted and all gates are assumed to be in Open state. Should be set for the hardware to start processing the gate control lists. During the toggle from 0 to 1, the gate control list processing starts only after the SSWL bit it set.

0: EST disabled

1: EST enabled

EST Extended Control register (ETH_MTLESTECR)

Address offset: 0x0C54

Reset value: 0x0000 0000

This register indicates the number of Overhead bytes for EST related scheduling.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVHD[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **OVHD[5:0]**: Overhead Bytes Value

This field indicates the fixed overhead for every packet to account for EST Scheduler Delay, IPG or EIPG, and Preamble bytes.

EST Status register (ETH_MTLESTSR)

Address offset: 0x0C58

Reset value: 0x0000 0000

This register provides the Status related to Enhancements to Scheduled Transmission (IEEE802.1Qbv).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGSN[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTRL[7:0]								SWOL	Res.	Res.	CGCE	HLBS	HLBF	BTRE	SWLC
r	r	r	r	r	r	r	r	r			rc_w1	r	r	rc_w1	rc_w1

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **CGSN[3:0]**: Current GCL slot number

Indicates the slot number of the GCL list. The slot number is a modulo 16 count of the GCL list loops executed so far. Even if a new GCL list is installed, the count is incremental.

Bits 15:8 **BTRL[7:0]**: BTR Error Loop Count

Provides the minimum count (N) for which the equation Current Time =< New BTR + (N × New Cycle Time) becomes true. N = 0b11111111 indicates the iterations exceeded the value of 128 and the hardware was not able to update New BTR to be equal to or greater than Current Time. Software intervention is needed to update the New BTR. Value cleared when BTRE field of this register is cleared.

Bit 7 **SWOL**: S/W owned list

When 0, indicates Gate Control list number “0” is owned by software and when 1, indicates Gate Control list “1” is owned by the software. Any reads/writes by the software (using indirect access through *EST Gate Control List register (ETH_MTLESTGCLCR)*) are directed to the list indicated by this value by default. The inverse of this value is treated as HWOL. R/W operations performed by hardware are directed to the list pointed by HWOL by default.

0: Inactive

1: Active

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CGCE**: Constant Gate Control Error

This error occurs when the list length (LLR) is 1 and the Cycle Time (CTR) is less than or equal to the programmed Time Interval (TI) value after the optional Left Shifting. This implies Gates are either always Closed or always Open based on the Gate Control values; the same effect can be achieved by other simpler (non TSN) programming mechanisms. Since the implementation does not support such a programming an error is reported.

Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

Bit 3 **HLBS**: Head-Of-Line Blocking due to Scheduling

Set when the frame is not able to win arbitration and get scheduled even after 4 iterations of the GCL. Indicates to software a potential programming error. The one hot encoded values of the Queue Numbers that are not able to make progress are indicated in the *EST Schedule Error register (ETH_MTLESTSCHER)*. Bit cleared when *EST Schedule Error register (ETH_MTLESTSCHER)* is all zeros.

0: Inactive

1: Active

Bit 2 **HLBF**: Head-Of-Line Blocking due to Frame Size

Set when HOL Blocking is noticed on one or more Queues as a result of none of the Time Intervals of gate open in the GCL being greater than or equal to the duration needed for frame size (or frame fragment size when preemption is enabled) transmission. The one hot encoded Queue numbers that are experiencing HLBF are indicated in the *EST Frame size Error register (ETH_MTLESTFSER)*. Additionally, the first Queue number that experienced HLBF along with the frame size is captured in *EST Frame size Capture register (ETH_MTLESTFSCR)*. HLBF bit is cleared when *EST Frame size Error register (ETH_MTLESTFSER)* is all zeros.

Bit 1 **BTRE**: BTR Error

When 1, indicates a programming error in the BTR of SWOL where the programmed value is less than current time. If the BTRL = 0b11111111, SWOL is not updated and the software should reprogram the BTR to a value greater than the current time and then set SSWL to reinitiate the switch to SWOL. Else if the value of BTRL < 0b11111111, SWOL is updated and this field indicates the number of iterations (of + CycleTime) taken by hardware to update the BTR to a value greater than the Current Time.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

Bit 0 **SWLC**: Switch to S/W owned list Complete

When 1, indicates the hardware has successfully switched to the SWOL, and the SWOL bit has been updated to that effect. Cleared when the SSWL of *EST Control register (ETH_MTLESTCR)* transitions from 0 to 1, or on a software write.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

EST Schedule Error register (ETH_MTLESTSCHER)

Address offset: 0x0C60

Reset value: 0x0000 0000

This register provides the one hot encoded queue numbers that are having the scheduling related error (timeout)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQN[3:0]			
												rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SEQN[3:0]**: Schedule Error Queue Number
The One Hot Encoded Queue Numbers that have experienced error/timeout described in HLBS field of *EST Status register (ETH_MTLESTSR)*.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

EST Frame size Error register (ETH_MTLESTFSER)

Address offset: 0x0C64

Reset value: 0x0000 0000

This register provides the one hot encoded queue numbers that are having the Frame Size related error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEQN[3:0]			
												rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **FEQN[3:0]**: Frame Size Error Queue Number
The One Hot Encoded Queue Numbers that have experienced error described in HLBF field of status register.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

EST Frame size Capture register (ETH_MTLESTFSCR)

Address offset: 0x0C68

Reset value: 0x0000 0000

This register captures the Frame Size and Queue Number of the first occurrence of the Frame Size related error. Up on clearing it captures the data of immediate next occurrence of a similar error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HBFQ[1:0]	
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HBFS[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **HBFQ[1:0]**: Queue Number of HLBF

Captures the binary value of the of the first Queue (number) experiencing HLBF error (see HLBF field of *EST Status register (ETH_MTLESTSR)*). Value once written is not altered by any subsequent queue errors of similar nature. Once cleared the queue number of the next occurring HLBF error is captured.

Cleared when *EST Frame size Error register (ETH_MTLESTFSER)* is all zeros.

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **HBFS[14:0]**: Frame Size of HLBF

Captures the Frame Size of the dropped frame related to queue number indicated in HBFQ field of this register. The content of this register should be considered invalid if this field is zero.

Cleared when *EST Frame size Error register (ETH_MTLESTFSER)* is all zeros.

EST Interrupt Enable register (ETH_MTLESTIER)

Address offset: 0x0C70

Reset value: 0x0000 0000

This register implements the Interrupt Enable bits for the various events that generate an interrupt. Bit positions have a 1 to 1 correlation with the status bit positions in *EST Status register (ETH_MTLESTSR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGCE	IEHS	IEHF	IEBE	IECC
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CGCE**: Interrupt Enable for CGCE

When set, this bit generates an interrupt when the Constant Gate Control Error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 3 **IEHS**: Interrupt Enable for HLBS

When set, this bit generates an interrupt when the head-of-line blocking due to a scheduling issue and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 2 **IEHF**: Interrupt Enable for HLBF

When set, this bit generates an interrupt when the head-of-line blocking due to Frame Size error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 1 **IEBE**: Interrupt Enable for BTR Error

When set, this bit generates an interrupt when the BTR error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 0 **IECC**: Interrupt Enable for Switch List

When set, this bit generates interrupt when the configuration change is successful and the hardware has switched to the new list. When reset, this event does not generate an interrupt.

EST Gate Control List register (ETH_MTLESTGCLCR)

Address offset: 0x0C80

Reset value: 0x0000 0000

This register provides the control information for reading/writing to the Gate Control lists.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADDR[5:0]						Res.	Res.	DBGB	DBGM	Res.	GCRR	R1W0	SRWO
		rw	rw	rw	rw	rw	rw			rw	rw		rw	rw	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDR[5:0]**: Gate Control List Address:

Condition: GCRR = 0

This bitfield provides the address (row number) of the gate control list at which the R/W operation has to be performed. By default the gate control list pointed by SWOL of *EST Status register (ETH_MTLESTSR)* is selected for R/W, however if the DBGM bit of this register is set, a debug mode access is given to read/write from DBGB.

Condition: GCRR = 1

By default the GCL related register set pointed by SWOL of *EST Status register (ETH_MTLESTSR)* is selected for R/W, however if the DBGM bit of this register is set, a debug mode access is given to R/W from DBGB. Lower 3 bits are only used in this mode, higher order bits are treated as don't care.

000: BTR Low (31:0)
001: BTR High (63:31)
010: CTR Low (31:0)
011: CTR High (39:32)
100: TER (30:0)
101: LLR (6:0)
Others: Reserved, must not be used

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **DBGB**: Debug Mode Bank Select

When set to 0, this bit indicates that R/W in Debug mode should be directed to Bank 0 (GCL0 and corresponding Time-related registers). When set to 1, it indicates that R/W in Debug mode should be directed to Bank 1 (GCL1 and corresponding Time-related registers). This value is used when DBGM is set and overrides by value of SWOL which is normally used.

0: Bank 0
1: Bank 1

Bit 4 **DBGM**: Debug Mode

When set to 1, this bit indicates R/W in Debug mode where the memory bank (for GCL and Time related registers) is explicitly provided by DBGB value, when set. When set to 0, SWOL bit is used to determine which bank to use.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GCRR**: Gate Control Related registers

When set to 1, this bit indicates the R/W access is directed to the GCL related registers (BTR, CTR, TER, LLR) whose address is provided by GCRA. When 0, it indicates R/W operations should be directed to GCL from the address provided by GCLA.

Bit 1 **R1W0**: Read 1, Write 0

1: Read operation

0: Write operation.

Bit 0 **SRWO**: Start Read/Write Operation

1: Indicates the start/progress of a Read/Write operation

Reset by hardware: Indicates the R/W operation has completed or an error has occurred (when bit 20 is set)

- Reads: Data can be read from *EST Gate Control List Data register (ETH_MTLESTGCLDR)* after this bit is reset.
- Writes: *EST Gate Control List Data register (ETH_MTLESTGCLDR)* should be programmed with write data before setting SRWO.

EST Gate Control List Data register (ETH_MTLESTGCLDR)

Address offset: 0x0C84

Reset value: 0x0000 0000

This register holds the read data or write data in case of reads and writes respectively.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GCD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GCD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **GCD[31:0]**: Gate Control Data

The data corresponding to the address selected in *EST Gate Control List register (ETH_MTLESTGCLCR)*. Used both for Read and Write operations.

FPE Frame Preemption Control Status register (ETH_MTLFPECSR)

Address offset: 0x0C90

Reset value: 0x0000 0000

This register provides the control information for reading/writing to the Gate Control lists.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PEC[3:0]				LBHT	Res.	Res.	Res.	Res.	Res.	AFSZ[1:0]	
				rw	rw	rw	rw	rw						rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRS**: Hold/Release Status

1: Indicates a Set-and-Hold-MAC operation was last executed and the pMAC is in Hold State.
0: Indicates a Set-and-Release-MAC operation was last executed and the pMAC is in Release State.

Bits 27:12 Reserved, must be kept at reset value.

Bits 11:8 **PEC[3:0]**: Preemption Classification

When set, this bit indicates the corresponding Queue must be classified as preemptable, when 0, the Queue is classified as express. When both EST (Qbv) and Preemption are enabled, Queue 0 is always assumed to be preemptable. When EST (Qbv) is enabled Queues categorized as preemptable here are always assumed to be in “Open” state in the Gate Control List.

Bit 7 **LBHT**: Level-based Hold Transition

When set, this bit enables the hold operation in the second row of the EST GCL if the hold request is set in the second GCL row for the first loop of the new GCL, even though the hold request is also set in the first GCL row. This is a deviation from the original behavior where a transition from the release request to the hold request is required.

When cleared, this bit enables the hold operation only on the transition from the release request to the hold request.

Bits 6:2 Reserved, must be kept at reset value.

Bits 1:0 **AFSZ[1:0]**: Additional Fragment Size

Used to indicate, in units of 64 bytes, the minimum number of bytes over 64 bytes required in non-final fragments of preempted frames. The minimum non-final fragment size is (AFSZ+1) × 64 bytes

FPE Frame Preemption Advance register (ETH_MTLFPEAR)

Address offset: 0x0C94

Reset value: 0x0000 0000

This register holds the Hold and Release Advance time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RADV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HADV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RADV[15:0]**: Release Advance

The maximum time in nanoseconds that can elapse between issuing a RELEASE to the MAC and the MAC being ready to resume transmission of preemptable frames, in the absence of there being any express frames available for transmission.

Bits 15:0 **HADV[15:0]**: Hold Advance

The maximum time in nanoseconds that can elapse between issuing a HOLD to the MAC and the MAC ceasing to transmit any preemptable frame that is in the process of transmission or any preemptable frames that are queued for transmission.

Tx queue x operating mode register (ETH_MTLTXQxOMR)

Address offset: 0x0D00 + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Queue x Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **TQS[4:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes. Queue size range from 256 bytes (TQS = 0b00000) to 8192 bytes (TQS = 0b11111).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **TTC[2:0]**: Transmit Threshold Control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit Queue Enable

This field is used to enable/disable the transmit queue 0.

00: Not enabled
01: Enable in AV mode
10: Enabled
Others: Reserved, must not be used.

Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit Store and Forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush Transmit Queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQ1OMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue x underflow register (ETH_MTLTXQxUR)

Address offset: 0x0D04 + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Queue x Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UFCNT OVF	UFFRMCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow Packet Counter

This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow Packet Counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue x debug register (ETH_MTLTXQxDR)

Address offset: 0x0D08 + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Queue x Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]		
									r	r	r		r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
										r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **STXSTSF[2:0]**: Number of Status Words in Tx Status FIFO of Queue

This field indicates the current number of status in the Tx Status FIFO of this queue.
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of Packets in the Transmit Queue

This field indicates the current number of packets in the Tx queue.
When the DTXSTS bit of *Operating mode register (ETH_MTLOMR)* register is set to 1, this field does not reflect the number of packets in the Transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx Status FIFO Full Status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

Bit 4 **TXQSTS**: MTL Tx Queue Not Empty Status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status
When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status
This field indicates the state of the Tx Queue Read Controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
- Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
- Reception of 802.3x Pause packet when PFC is disabled

Tx queue x ETS status register (ETH_MTLTXQxESR)

Address offset: 0x0D14 + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

The Queue x ETS Status register provides the average traffic transmitted in Queue x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ABS[23:0]**: Average Bits per Slot
This field contains the average transmitted bits per slot.

Queue x interrupt control status register (ETH_MTLQxICSR)

Address offset: 0x0D2C + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue x interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
						rw	rw							rw	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable

When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFIS**: Receive Queue Overflow Interrupt Status

This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ABPSIE**: Average Bits Per Slot Interrupt Enable

When this bit is set, the MAC asserts the eth_sbd_intr_it interrupt when the average bits per slot status is updated.

When this bit is cleared, the interrupt is not asserted for such an event.

Bit 8 **TXUIE**: Transmit Queue Underflow Interrupt Enable

When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ABPSIS**: Average Bits Per Slot Interrupt Status

When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit.

Note: This bit is automatically set after 10 ms in 1000 Mbps mode (100 ms in 100 Mbps), once the reset is released, regardless of the Ethernet controller configuration.

Bit 0 **TXUNFIS**: Transmit Queue Underflow Interrupt Status

This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue x operating mode register (ETH_MTLRXQxOMR)

Address offset: 0x0D30 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Receive operating Mode register establishes the Receive queue operating modes and command.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD [2]
								rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFD[1:0]		Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_ TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **RQS[3:0]**: Receive Queue Size

This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The size of the queues is (RQS + 1) × 256 bytes.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **RFD[2:0]**: Threshold for Deactivating Flow Control (in Half-duplex and Full-duplex modes)

These bits control the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation:

0: Full minus 1 Kbyte

1: Full minus 1.5 Kbyte

2: Full minus 2 Kbytes

3: Full minus 2.5 Kbytes

4: Full minus 3 Kbytes

5: Full minus 3.5 Kbytes

Other values: reserved

The deassertion is effective only after flow control is asserted.

Note: The value must be programmed in such a way to make sure that the threshold is a positive number.

To have the flow control activated for a fill level of 3 Kbytes and deactivated when the fill level is 1 Kbyte, program:

RFA[2:0] = 0 (full – 1 Kbyte)

RFD[2:0] = 4 (full – 3 Kbytes)

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **RFA[2:0]**: Threshold for Activating Flow Control (in Half-duplex and Full-duplex)

These bits control the threshold (fill-level of Rx queue) at which the flow control is activated:
For more information on encoding for this field, see RFD[2:0].

Bit 7 **EHFC**: Enable Hardware Flow Control

When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.

Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):

00: 64

01: 32

10: 96

11: 128

The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.

This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)

Address offset: 0x0D34 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed Packet Counter Overflow Bit

When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter

This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.

This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit

When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter

This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue x debug register (ETH_MTLRXQxDR)

Address offset: 0x0D38 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Receive Debug register gives the debug status of various blocks related to the Receive queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRXQ[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
										r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue

This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbyte/16 bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status

This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State

This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status

When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Rx queue x control register (ETH_MTLRXQxCR)

Address offset: 0x0D3C + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RXQ_FRM_ARBIT**: Receive Queue Packet Arbitration

When this bit is set, the Ethernet peripheral drives the packet data to the DMA such that the entire packet data of currently-selected queue is transmitted before switching to other queue.
When this bit is reset, the Ethernet peripheral drives the packet data to the DMA such that the following amount of data of currently-selected queue is transmitted before switching to other queue:

– PBL amount of data

or

– complete data of a packet

The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet peripheral drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).

Bits 2:0 **RXQ_WEGT[2:0]**: Receive Queue Weight

This field indicates the weight assigned to the Rx queue x. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle.
This field needs to be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on.

Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the Rx queue x control register (ETH_MTLRXQxCR) must be programmed before the Rx queue x operating mode register (ETH_MTLRXQxOMR).

Tx queue x ETS control register (ETH_MTLTXQxECR)

Address offset: 0x0D10 + 0x40 * x, (x = 1 to 3)

Reset value: 0x0000 0000

The Queue ETS Control register controls the enhanced transmission selection operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
									rw	rw	rw	rw	rw		

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SLC[2:0]**: Slot Count

If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 us duration) over which the average transmitted bits per slot, provided in *Tx queue x ETS status register (ETH_MTLTXQxESR)*, need to be computed for Queue x. The encoding is as follows:
000: 1 Slot
001: 2 Slots
010: 4 Slots
011: 8 Slots
100: 16 Slots
101 to 111: Reserved, must not be used.

Bit 3 **CC**: Credit Control

When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel x. The credit accumulates even when there is no packet waiting in Channel x and another channel is transmitting.
When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel x. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated.

Bit 2 **AVALG**: AV Algorithm

When Queue x is programmed for AV, this field configures the scheduling algorithm for this queue:
This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue x traffic. When reset, strict priority is selected.

Bits 1:0 Reserved, must be kept at reset value.

Tx queue x quantum weight register (ETH_MTLTXQxQWR)

Address offset: 0x0D18 + 0x40 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register contains the weights for the Weighted Round Robin (WRR) or IdleSlopeCredit for the queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ISCQW[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **ISCQW[13:0]**: IdleSlopeCredit or Weights

idleSlopeCredit

When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue x (x = 1 to 3). This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode.

Weights

If WRR algorithm is enabled for queue x generic traffic (not AVB), this field contains the weight for queue x. The maximum value is 0x64. Bits [13:7] must be written to zero.

Note: In multiple Queue configuration this field must be programmed to some non-zero value when multiple queues are enabled, or a single queue other than Queue 0 is enabled. This field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both Receive and Transmit. In Receive, the register is Operating mode register (ETH_MTLOMR).

The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time.

Tx queue x send slope credit register (ETH_MTLTXQxSSCR)

Address offset: 0x0D1C + 0x40 * x, (x = 1 to 3)

Reset value: 0x0000 0000

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SSC[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **SSC[13:0]**: sendSlopeCredit Value

When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for queue x. This is the rate of change of credit in bits per cycle (40 ns and 8 ns for 100 Mbps and 1000 Mbps respectively) when the credit decreases. The software should program this field with computed credit in bits per cycle scaled by 1 024. The maximum value is portTransmitRate, that is 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when queue x is selected for transmission.

Tx Queue x hiCredit register (ETH_MTLTXQxHCR)

Address offset: 0x0D20 + 0x40 * x, (x = 1 to 3)

Reset value: 0x0000 0000

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **HC[28:0]**: hiCredit Value

This field contains the hiCredit value required for the credit-based shaper algorithm when the AV feature is enabled. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024.

The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16 384 bytes or 131 072 bits). The value to be specified is 131 072 ×1 024 = 134 217 728 or 0x0800 0000.

Tx queue x loCredit register (ETH_MTLTXQxLCR)

Address offset: 0x0D24 + 0x40 * x, (x = 1 to 3)

Reset value: 0x0000 0000

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **LC[28:0]**: loCredit Value

When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024. The maximum value is maxFrameSize transmitted from this queue, that is, 16 384 × 8 × 1 024 = 134 217 728 or 0x0800 0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800 0000.

Ethernet MTL register map and reset values

Table 885. ETH_MTL register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTXSTS	Res.
	Reset value																							0	0		0	0			0	0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESTIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q3IS	Q2IS	Q1IS	Q0IS
	Reset value														0															0	0	0	0
0x0C24 - 0x0C2C	Reserved																																
0x0C30	ETH_MTLRXQDMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1DDMACH	Res.	Res.	Res.	Q1MDMACH	Res.	Res.	Res.	Q0DDMACH	Res.	Res.	Res.	Q0MDMACH
	Reset value																				0				0				0				0
0x0C34 - 0x0C3C	Reserved																																
0x0C40	ETH_MTLTBSCR	LEOS[23:0]																								LEGOS[2:0]				Res.	Res.	LEOV	ESTM
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0			0	0
0x0C50	ETH_MTLESTCR	PTOV[7:0]								CTOV[11:0]												Res.	TILS[2:0]			LCSE[1:0]		DFBS	DDBF	Res.	Res.	SSWL	EEST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0			0	0
0x0C54	ETH_MTLESTECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVHD[5:0]				
	Reset value																											0	0	0	0	0	0
0x0C58	ETH_MTLESTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGSN[3:0]				BTRL[7:0]								SWOL	Res.	Res.	CGCE	HLBS	HLBF	BTRE	SWLC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x0C5C	Reserved																																
0x0C60	ETH_MTLESTSCHER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQN[3]	SEQN[2]	SEQN[1:0]	
	Reset value																													0	0	0	0

Table 885. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C64	ETH_ MTLESTFSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEQN[3]	FEQN[2]	FEQN[1:0]	
	Reset value																													0	0	0	0
0x0C68	ETH_ MTLESTFSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		HBFQ[1]	HBFQ[0]	Res.	HBFS[14:0]														
	Reset value															0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C6C	Reserved																																
0x0C70	ETH_MTLESTIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGCE	IEHS	IEHF	IEBE	IECC
	Reset value																												0	0	0	0	0
0x0C74 - 0x0C7C	Reserved																																
0x0C80	ETH_ MTLESTGCLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	ADDR[5:0]						Res.	Res.	DBGB	DBGM	Res.	GCRR	R1W0	SRWO
	Reset value																			0	0	0	0	0	0			0	0		0	0	0
0x0C84	ETH_ MTLESTGCLDR	GCD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C88- 0x0C8C	Reserved																																
0x0C90	ETH_ MTLFPECSR	Res.	Res.	Res.	HRS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	PEC[3]	PEC[2]	PEC[1:0]		LBHT	Res.	Res.	Res.	Res.	Res.	AFSZ[1:0]	
	Reset value				0																	0	0	0	0	0						0	0
0x0C94	ETH_MTLFPEAR	RADV[15:0]																HADV[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C98 - 0x0CFC	Reserved																																
0x0D00	ETH_ MTLTXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x0D04	ETH_ MTLTXQ0UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_ MTLTXQ0DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.		PTXQ[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0

Table 885. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D0C	Reserved																																
0x0D10	Reserved																																
0x0D14	ETH_ MTLTXQ0ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D18	ETH_ MTLTXQ0QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0D1C - 0x0D24	Reserved																																
0x0D28	Reserved																																
0x0D2C	ETH_MTLQ0ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0						0	0
0x0D30	ETH_ MTLRXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D34	ETH_ MTLRXQ0MPOCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D38	ETH_ MTLRXQ0DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0		0	0	0
0x0D3C	ETH_ MTLRXQ0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0
0x0D40	ETH_ MTLTXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x0D44	ETH_ MTLTXQ1UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 885. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D48	ETH_ MTLTXQ1DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D4C	Reserved																																
0x0D50	ETH_ MTLTXQ1ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
	Reset value																										0	0	0	0	0		
0x0D54	ETH_ MTLTXQ1ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D58	ETH_ MTLTXQ1QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D5C	ETH_ MTLTXQ1SSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D60	ETH_ MTLTXQ1HCR	Res.	Res.	Res.	HC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D64	ETH_ MTLTXQ1LCR	Res.	Res.	Res.	LC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D68	Reserved																																
0x0D6C	ETH_MTLQ1ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0
0x0D70	ETH_ MTLRXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D74	ETH_MTLRXQ1MP OCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D78	ETH_ MTLRXQ1DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0

Table 885. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D7C	ETH_ MTLRXQ1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0
0x0D80	ETH_ MTLTXQ2OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x0D84	ETH_ MTLTXQ2UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D88	ETH_ MTLTXQ2DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]		Res.		PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D8C	Reserved																																
0x0D90	ETH_ MTLTXQ2ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
	Reset value																										0	0	0	0	0		
0x0D94	ETH_ MTLTXQ2ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D98	ETH_ MTLTXQ2QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D9C	ETH_ MTLTXQ2SSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DA0	ETH_ MTLTXQ2HCR	Res.	Res.	Res.	HC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DA4	ETH_ MTLTXQ2LCR	Res.	Res.	Res.	LC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DA8	Reserved																																
0x0DAC	ETH_MTLQ2ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0						0	0

Table 885. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0DC0	ETH_ MTLTXQ3OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x0DC4	ETH_ MTLTXQ3UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0DC8	ETH_ MTLTXQ3DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]		Res.		PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0										0	0	0	0	0	0	0
0x0DCC	Reserved																																
0x0DD0	ETH_ MTLTXQ3ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
	Reset value																										0	0	0	0	0		
0x0DD4	ETH_ MTLTXQ3ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DD8	ETH_ MTLTXQ3QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DDC	ETH_ MTLTXQ3SSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DE0	ETH_ MTLTXQ3HCR	Res.	Res.	Res.	HC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DE4	ETH_ MTLTXQ3LCR	Res.	Res.	Res.	LC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DE8	Reserved																																
0x0DEC	ETH_MTLQ3ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

77.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 0000

The MAC Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN		SARC[2:0]		IPC		IPG[2:0]		GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

010:the MAC inserts the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

011: the MAC replaces the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

110: the MAC inserts the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets

111: the MAC replaces the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

Others: Reserved, must not be used.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

000: 96 bit times

001: 88 bit times

010: 80 bit times

...

111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode.

In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 886: Giant Packet Status based on S2KP and JE Bits*.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 887: Packet Length based on the CST and ACS bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see Table 887: Packet Length based on the CST and ACS bits.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 **BE**: Packet Burst Enable

When this bit is set, the MAC allows packet bursting during transmission in the GMII Half-duplex mode.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 886: Giant Packet Status based on S2KP and JE Bits*.

Bit 15 **PS**: Port Select

This bit selects the Ethernet line speed.

0: For 1000 Mbps operations

1: For 10 or 100 Mbps operations

In 10 or 100 Mbps operations, this bit, along with Bit FES, selects the exact line speed.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbps mode:

0: 10 Mbps

1: 100 Mbps

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at GMII or MII. The GMII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the GMII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range $0 <= r < 2^k$.

This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral Check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.

If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble Length for Transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved, must not be used

Bit 1 **TE**: Transmitter Enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver Enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface.

Table 886 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 886. Giant Packet Status based on S2KP and JE Bits(1)

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1

1. For all other combinations, the Giant Packet status is 0.

Table 887 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 887. Packet Length based on the CST and ACS bits

Received Packet Length	CST	ACS	FCS Stripping Done
< 1,536	x	0	No
	x	1	Yes (for Ethernet packets)
≥ 1,536	0	x	No
	1	x	Yes (for Type packets)

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GPSL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **APDIM**: ARP Packet Drop if IP Address Mismatch

When this bit is set, Packet for which Target Protocol Address does not match IPv4 address is dropped in the MTL layer.
When this bit is reset, when target Protocol Address does not match, the packet is forwarded to MTL maintaining backward compatibility.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in *Operating mode configuration register (ETH_MACCR)*, gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in *Operating mode configuration register (ETH_MACCR)* together as minimum IPG greater than 96 bit times in steps of 8 bit times.
When this bit is reset, the MAC ignores EIPG field and interprets IPG field in *Operating mode configuration register (ETH_MACCR)* as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes.

When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.

When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.

For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.

When this bit is reset, the receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog and jabber timeout register (ETH_MACWJBTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The watchdog and jabber timeout register controls the watchdog timeout for received packets and the jabber timeout limit for transmitted packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PJE	Res.	Res.	Res.	Res.	JTO[3:0]			
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
							rw					rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PJE**: Programmable Jabber Enable

When this bit is set and the JD bit of the *Operating mode configuration register (ETH_MACCR)* is reset, the JTO[3:0] bitfield is used as jabber timeout limit for transmitted packets.

When this bit is cleared, the jabber timeout for a received packet is controlled by setting the JD and JE bits of the *Operating mode configuration register (ETH_MACCR)*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **JTO[3:0]**: Jabber Timeout

When the PJE bit is set and the JD bit of the *Operating mode configuration register (ETH_MACCR)* is cleared, this bitfield is used as jabber timeout limit for transmitted packets. If the length of a transmitted packet exceeds the value of this bitfield, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes

0x1: 3 Kbytes

0x2: 4 Kbytes

0x3: 5 Kbytes

..

0xC: 14 Kbytes

0xD: 15 Kbytes

0xE: 16383 Bytes

0xF: Reserved, must not be used

Note: When the PJE bit is set, the value in this bitfield must be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and terminated.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in *Operating mode configuration register (ETH_MACCR)* register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog Timeout

When the PWE bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes
0x1: 3 Kbytes
0x2: 4 Kbytes
0x3: 5 Kbytes
..
0xC: 14 Kbytes
0xD: 15 Kbytes
0xE: 16383 Bytes
0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this bitfield should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits
This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits
This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag Control register (ETH_MACVTCR)

Address offset: 0x0050

Reset value: 0x0000 0000

This register is the redefined format of the MAC VLAN Tag Register. It is used for indirect addressing. It contains the address offset, command type and Busy Bit for CSR access of the Per VLAN Tag registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFS[1:0]		CT	OB
												rw	rw	rw	rw

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.
When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison.
When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.
When this bit is reset, the VLAN Hash Match operation is not performed.

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive

This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check

When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match

When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN

When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable

When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison

When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:2 **OFS[1:0]**: Offset

This field holds the address offset of the MAC VLAN Tag Filter Register which the application is trying to access.
00:ETH_MACVTDR holds MAC VLAN Tag Filter0 content
01:ETH_MACVTDR holds MAC VLAN Tag Filter1 content
10:ETH_MACVTDR holds MAC VLAN Tag Filter2 content
11:ETH_MACVTDR holds MAC VLAN Tag Filter3 content

Bit 1 **CT**: Command Type

This bit indicates if the current register access is a read or a write.
When set, it indicate a read operation. When reset, it indicates a write operation.

Bit 0 **OB**: Operation Busy

This bit is set along with a read or write command for initiating the indirect access to per VLAN Tag Filter register. This bit is reset when the read or write command to per VLAN Tag Filter indirect access register is complete. The next indirect register access can be initiated only after this bit is reset.
During a write operation, the bit is reset only after the data has been written into the Per VLAN Tag register.
During a read operation, the data should be read from the MAC_VLAN_Tag_Data register only after this bit is reset.

VLAN tag data register (ETH_MACVTDR)

Address offset: 0x0054

Reset value: 0x0000 0000

This register contains VLAN Tag filter 0..3 control information, as per OFS[1:0] bitfield setting in ETH_MACVTCR register. This register holds the read/write data for Indirect Access of the Per VLAN Tag registers. During the read access, this field contains valid read data only after the OB bit is reset.
During the write access, this field should be valid prior to setting the OB bit in the MAC_VLAN_Tag_Ctrl Register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DMACHN	DMACHEN	Res.	Res.	Res.	ERIVLT	ERSVLM	DOVLTC	ETV	VEN
						rw	rw				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DMACHN**: DMA Channel Number

The DMA Channel number to which the VLAN Tagged Frame is to be routed if it passes this VLAN Tag Filter is programmed in this field.
If the Routing based on VLAN Tag Filter is not necessary, this field need not be programmed.

Bit 24 **DMACHEN**: DMA Channel Number Enable

This bit is the Enable for the DMA Channel Number value programmed in the DMACH field. When this bit is reset, the Routing does not occur based on VLAN Filter result. The frame is routed based on DA-based DMA Channel Routing.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ERIVLT**: Enable Inner VLAN Tag Comparison

This bit is valid only when Double VLAN Tag enable of the Filter is set.
When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present).
When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present).

Bit 19 **ERSVLM**: Enable S-VLAN Match for received Frames

This bit is valid only when VLAN Tag Enable of the Filter is set.
When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets.
When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type =0x8100) packets.

Bit 18 **DOVLTC**: Disable VLAN Type Comparison

This bit is valid only when VLAN Tag Enable of the Filter is set.
When this bit is set, the MAC does not check whether the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when the VLAN Tag type is similar to the one specified by the Enable S-VLAN Match for received frames bit.

Bit 17 **ETV**: 12-bit or 16-bit VLAN comparison

This bit is valid only when VEN of the Filter is set.
When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared to the corresponding field in the received VLAN-tagged packet.
0: 16-bit VLAN
1: 12-bit VLAN

Bit 16 **VEN**: VLAN Tag Enable

This bit is used to enable or disable the VLAN Tag.
When this bit is set, the MAC compares the VLAN Tag of received packet with the VLAN Tag ID.
When this bit is reset, no comparison is performed irrespective of the programming of the other fields.

Bits 15:0 **VID[15:0]**: VLAN Tag ID

This field holds the VLAN Tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the VTHM bit of *VLAN tag Control register (ETH_MACVTCR)* register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of *VLAN tag Control register (ETH_MACVTCR)* register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLHT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUSY	RDWR	Res.	Res.	Res.	Res.	ADDR1	ADDR0	Res.	Res.	CBTI	VLTI	CSVL	VLP	VLC[1:0]	
r	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BUSY**: Busy

This bit indicates the status of the read/write operation of indirect access to the queue/channel specific VLAN inclusion register.

For write operation write to a register is complete when this bit is reset. For read operation the read data is valid when the bit is reset.

No further writes are allowed to this register when this bit is set. This does not have any effect when CBTI is reset

Bit 30 **RDWR**: Read write control

This bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register.

When set indicates write operation and when reset indicates read operation

This has no effect when CBTI is reset

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **ADDR[1:0]**: Address

This field selects one of the queue/channel specific VLAN Inclusion register for read/write access

This does not have any effect when CBTI is reset

00: VLAN tag for insertion in the Transmit packets from Tx Queue 0

01: VLAN tag for insertion in the Transmit packets from Tx Queue 1

10: VLAN tag for insertion in the Transmit packets from Tx Queue 2

11: VLAN tag for insertion in the Transmit packets from Tx Queue 3

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CBTI**: Channel based tag insertion

When this bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. This overrides VLTI, VLP, VLC and VLT fields of this register

When this bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register.

When reset outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

VLAN inclusion register [alternate] (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The Tx Queue x VLAN Tag Inclusion register contains the VLAN tag for insertion in the Transmit packets from Tx Queue x. It also contains the VLAN tag insertion controls. This is an indirect register accessible when the CBTI bit is set in *VLAN inclusion register (ETH_MACVIR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSVL	Res.	Res.	Res.
												rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue 0 flow control register (ETH_MACQ0TXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
								rw	rw	rw	rw			rw	rw

Bits 31:16 **PT[15:0]**: Pause Time

This field holds the value to be used in the Pause Time field in the Tx control packet. I

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets.
When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface.

This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-duplex mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-duplex mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

Full-Duplex mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-duplex mode: When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

Rx Queue control register (ETH_MACRXQCR)

Address offset: 0x0094

Reset value: 0x0000 0000

The Receive Queue Control register controls the routing of unicast and multicast packets that fail the Destination or Source address filter to the Rx queues

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
						rw	rw							rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VFFQ**: VLAN Tag Filter Fail Packets Queue

This field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and UFFQE/MFFQE not enabled) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 16 **VFFQE**: VLAN Tag Filter Fail Packets Queuing Enable

When this bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed to the Rx Queue Number programmed in the VFFQ. When this bit is reset, the tagged packets which fail the Destination or Source Address filter or fail the VLAN tag filter are routed based on other routing options. This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

0: Disabled

1: Enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MFFQ**: Multicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the MFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 8 **MFFQE**: Multicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the MFFQ.

When this bit is reset, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **UFFQ**: Unicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the UFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 0 **UFFQE**: Unicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the UFFQ.

When this bit is reset, the Unicast packets which fail the Destination or Source Address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Rx queue control 0 register (ETH_MACRXQC0R)

Address offset: 0x00A0

Reset value: 0x0000 0000

The Receive Queue Control 0 register controls the queue management in the MAC receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **RXQ1EN[1:0]**: Receive Queue 1 Enable

This field indicates whether Rx queue 1 is enabled for AV or Generic traffic.

00: Not enabled

01: Queue 1 enabled for AV

10: Queue 1 enabled for Generic traffic

11: Reserved, must not be used

Bits 1:0 **RXQ0EN[1:0]**: Receive Queue 0 Enable

This field indicates whether Rx queue 0 is enabled for AV or Generic traffic.

00: Not enabled

01: Queue 0 enabled for AV

10: Queue 0 enabled for Generic traffic

11: Reserved, must not be used

Rx queue control 1 register (ETH_MACRXQC1R)

Address offset: 0x00A4

Reset value: 0x0000 0000

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV and untagged packets to the Rx queues.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TBRQE	OMCBCQ	Res.	FPRQ2	FPRQ1	FPRQ0	TPQC[1:0]		TACPQE	MCBCQEN	Res.	MCBCQ[2:0]		
		rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ2	AVCPQ1	AVCPQ0
	rw	rw	rw						rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TBRQE**: Type Field Based Rx Queuing Enable

When this bit is set, it enables Type field based Rx queuing where the Type field of received packet is compared with the programmed TYP field in ETH_MAC_TMRQR register and if a match occurs the packet is routed to the corresponding TMRQ field.

Bit 28 **OMCBCQ**: Overriding MC-BC queue priority select

1: Priority of MCBCQ is reduced and the received packet is first routed to PTPQ, AVCPQ, depending on packet type.
0: Received Multicast/Broadcast packet is routed to MCBCQ.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **FPRQ[2:0]**: Frame Preemption Residue Queue

This field holds the Rx queue number to which the residual preemption frames must be forwarded. Preemption frames that are tagged and pass the SA/DA/VLAN filtering are routed based on PSRQ and all other frames are treated as residual frames and is routed to the queue number mentioned in this field. Queue 0 is used as a default queue for express frames, so this field cannot be programmed to a value 0.
000: Reserved, must not be used (Rx Queue 0)
001: Rx queue 1
Others: Reserved, must not be used

Bits 23:22 **TPQC[1:0]**: Tagged PTP over Ethernet Packets Queuing Control

This field controls the routing of the VLAN tagged PTPoE packets.
00: VLAN tagged PTPoE packets are routed as generic VLAN tagged packet (based on PSRQ for only non-AV enabled Rx queues).
01: VLAN tagged PTPoE packets are routed to Rx Queue specified by PTPQ field (this Rx Queue can be enabled for AV or non-AV traffic)
10: VLAN tagged PTPoE packets are routed to only AV enabled Rx queues based on PSRQ.
11: Reserved

Bit 21 **TACPQE**: Tagged AV Control Packets Queuing Enable
When set, the MAC routes the received Tagged AV control packets to the Rx queue specified by AVCPQ field.
When reset, the MAC routes the received Tagged AV control packets based on the tag priority matching the PSRQ fields in ETH_MACRXQC2R register.
This field is reserved if Rx queues are not selected for AV feature.

Bit 20 **MCBCQEN**: Multicast and Broadcast Queue Enable
This bit specifies that Multicast or Broadcast packets routing to the Rx queue is enabled and the Multicast or Broadcast packets must be routed to Rx queue specified in MCBCQ field.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **MCBCQ[2:0]**: Multicast and Broadcast Queue
This field specifies the Rx queue onto which Multicast or Broadcast packets are routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Multicast or Broadcast packets.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **UPQ[2:0]**: Untagged Packet Queue
This field indicates the Rx queue to which Untagged Packets are to be routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Untagged Packets.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used

Bits 11:7 Reserved, must be kept at reset value.

Bits 6:4 **PTPQ[2:0]**: PTP Packets Queue
This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used
When the AV8021ASMEN bit of *Timestamp control Register (ETH_MACTSCR)* is set, only untagged PTP over Ethernet packets are routed on an Rx queue. If the bit is not set, then based on programming of TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx Queue.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AVCPQ[2:0]**: AV Untagged Control Packets Queue
This field specifies the Receive queue on which the received AV untagged control packets are routed:
000: Receive Queue 0
001: Receive Queue 1
Others: Reserved, must not be used
The AV tagged control and data packets are routed based on PSRQ field in the Transmit Flow Control Register of corresponding queue.
This field is reserved if Receive queues are not selected for AV feature.

Rx queue control 2 register (ETH_MACRXQC2R)

Address offset: 0x00A8

Reset value: 0x0000 0000

This register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queue 0 and 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSRQ1[7:0]								PSRQ0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **PSRQ1[7:0]**: Priorities Selected in the Receive Queue 1

This field defines the priorities assigned to Rx queue 1. All packets with priorities that match the values set in this field are routed to Rx queue 1.

For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Bits 7:0 **PSRQ0[7:0]**: Priorities Selected in the Receive Queue 0

This field defines the priorities assigned to Rx queue 0. All packets with priorities that match the values set in this field are routed to Rx queue 0.

For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFRIS	MFTIS	MDIOIS	FPEIS	Res.
											r	r	rc_r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIIIS
	rc_r	rc_r	rc_r		r	r	r			r	r	r			r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **MFRIS**: MMC FPE Receive Interrupt Status

This bit is set high when an interrupt is generated in the *MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 19 **MFTIS**: MMC FPE Transmit Interrupt Status

This bit is set high when an interrupt is generated in the *MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 18 **MDIOIS**: MDIO Interrupt Status

This bit indicates an interrupt event after the completion of MDIO operation. To reset this bit, the application has to read this bit (or write it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* register is set) register is set.

Bit 17 **FPEIS**: Frame Preemption Interrupt Status

This bit indicates an interrupt event during the operation of Frame Preemption (Bits[19:16] of *FPE control and status register (ETH_MACFPECSR)* is set). To reset this bit, the application must clear the event in *FPE control and status register (ETH_MACFPECSR)* that has caused the Interrupt.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive Status Interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit Status Interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp Interrupt Status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

- The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.
- There is an overflow in the Seconds register.
- The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the *Timestamp status register (ETH_MACTSSR)*.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC Interrupt Status

This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI Interrupt Status

This bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of *LPI control and status register (ETH_MACLCSR)* is read.

Bit 4 **PMTIS**: PMT Interrupt Status

This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the *PMT control status register (ETH_MACPCSR)*.

Bit 3 **PHYIS**: PHY Interrupt

This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIS**: RGMII Interrupt Status

This bit is set by any change in value of the Link Status of RGMII interface (LNKSTS bit in *PHYIF control status register (ETH_MACPHYCSR)* register). This bit is cleared when the ETH_MACPHYCSR register is read.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	FPEIE	Res.
													rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	RGSMIIIE
	rw	rw	rw							rw	rw	rw			rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **MDIOIE**: MDIO Interrupt Enable
When this bit is set, it enables the assertion of the interrupt when MDIOIS field is set in the *Interrupt status register (ETH_MACISR)*.

Bit 17 **FPEIE**: Frame Preemption Interrupt Enable
When this bit is set, it enables the assertion of the interrupt when FPEIS field is set in the *Interrupt status register (ETH_MACISR)*.

Bit 16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 12 **TSIE**: Timestamp Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 4 **PMTIE**: PMT Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 3 **PHYIE**: PHY Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIE**: RGMII Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RGSMIIIS bit in *Interrupt status register (ETH_MACISR)*.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							rc_r			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive Watchdog Timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after the first collision and the packet transmission is aborted.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode).

This bit is not valid if the Underflow error occurs.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)* and the DC bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000 Mbps mode or when Jumbo packet is enabled).

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TJT**: Transmit Jabber Timeout

This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	rc_r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset
When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wake-up FIFO Pointer
This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wake-up Packet Forwarding Enable
When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.
The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.
Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast
When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wake-up Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wake-up Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACRWKPFR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACRWKPFR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MACRWKPFR[31:0]**: Remote wake-up packet filter

Refer to *Table 851*, *Table 852* and *Table 853* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.
After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.
When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.
If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQxOMR, when the MAC is in the LPI mode, it exits the LPI mode.
When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 **PLSEN**: PHY Link Status Enable

This bit enables the link status received on the RGMII Receive paths to be used for activating the LPI LS TIMER.
When this bit is set, the MAC uses the link-status bits of the ETH_MACPHYCSR register and the PLS bit for the LPI LS Timer trigger. When this bit is reset, the MAC ignores the link-status bits of the ETH_MACPHYCSR register and takes only the PLS bit.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.
When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.
This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **LST[9:0]**: LPI LS Timer

This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer

This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of *LPI control and status register (ETH_MACLCSR)* register is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPIET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **LPIET[19:0]**: LPI Entry Timer

This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

PHYIF control status register (ETH_MACPHYCSR)

Address offset: 0x00F8

Reset value: 0x0000 0000

The PHY Interface Control and Status register indicates the status signals received by the RGMII interface from the PHY.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNKSTS	LNKSPEED[1:0]		LNKMOD
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUD	TC
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **LNKSTS**: Link Status

This bit indicates whether the link is up or down.

0: Link down

1: Link up

Bits 18:17 **LNKSPEED[1:0]**: Link Speed

This bit indicates the current speed of the link:

00: 2.5 MHz

01: 25 MHz

10: 125 MHz

11: Reserved, must not be used

Bit 16 **LNKMOD**: Link Mode

This bit indicates the current operating mode of the link:

0: Half-duplex mode

1: Full-duplex mode

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LUD**: Link Up or Down

This bit indicates whether the link is up or down during transmission of configuration in the RGMII interface.
0: Link Down
1: Link Up

Bit 0 **TC**: Transmit Configuration in RGMII

When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII port. When this bit is reset, no such information is driven to the PHY (see *Section 77.6.3: Reduced media independent interface (RMII)*).

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 1B53

The version register identifies the version of the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC Transmit Packet Controller Status

This field indicates the state of the MAC Transmit Packet Controller module:

00: Idle state

01: Waiting for one of the following:

– Status of the previous packet

– IPG or backoff period to be over

10: Generating and transmitting a Pause control packet (in Full-duplex mode)

11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC GMII or MII Transmit Protocol Engine Status

When this bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC Receive Packet Controller FIFO Status

When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0 **RPESTS**: MAC GMII or MII Receive Protocol Engine Status

When this bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 0 register (ETH_MACHWF0R)

Address offset: 0x011C

Reset value: 0x0E0D 73F7

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	r	r	r			r	r	r	r	r	r	r	r	r	r

Bits 31:28 **ACTPHYSEL[3:0]**: Active PHY selected
When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
0x0: GMII or MII
0x1: RGMII
0x2: SGMII
0x3: TBI
0x4: RMII
0x5: RTBI
0x6: SMII
Others: Reserved, must not be used

Bit 27 **SAVLANINS**: Source Address or VLAN Insertion Enable
This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25 **TSSTSSEL[1:0]**: Timestamp System Time Source
This bit indicates the source of the Timestamp system time:
01: Internal
10: External
11: Both
00: Reserved, must not be used
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24 **MACADR64SEL**: MAC Addresses 64-127 Selected
This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23 **MACADR32SEL**: MAC Addresses 32-63 Selected
This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18 **ADDMACADRSEL[4:0]**: MAC Addresses 1-31 Selected
This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RXCOESEL**: Receive Checksum Offload Enabled
This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXCOESEL**: Transmit Checksum Offload Enabled
This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected

Bit 13 **EEESEL**: Energy Efficient Ethernet Enabled
This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12 **TSSEL**: IEEE 1588-2008 Timestamp Enabled
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ARPOFFSEL**: ARP Offload Enabled
This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8 **MMCSEL**: RMON Module Enable
This bit is set to 1 when the Enable MAC management counters (MMC) option is selected

Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbps Support
This bit is set to 1 when 1000 Mbps is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbps Support
This bit is set to 1 when 10/100 Mbps is selected as operating mode.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1114 59A5

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r	r		r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					SPRAM	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bit 23 **POUOST**: One Step for PTP over UDP/IP Feature Enable
This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **RAVSEL**: Rx Side Only AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19 **DBGMEMA**: DMA Debug Registers Enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP Segmentation Offload Enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split Header Feature Enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB Feature Enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved, must not be used

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register Enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP Offload Enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-Step Timestamping Enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size

This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011 to 11111: Reserved, must not be used

Bit 5 **SPRAM**: Single Port RAM Enable

This bit is set to 1 when the Use single port RAM feature is selected.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size

This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100 to 11111: Reserved, must not be used

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x420C 10C1

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]	
	r	r	r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
r	r	r	r			r	r	r	r			r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101 to 111: Reserved, must not be used

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS Outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101 to 111: Reserved, must not be used

Bits 23:22 **TDCSZ[1:0]**: Tx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA Transmit Channels
This field indicates the number of DMA Transmit channels:
0000: 1 DMA Tx Channel
0001: 2 DMA Tx Channels
..
0111: 8 DMA Tx

Bits 17:16 **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA Receive Channels
This field indicates the number of DMA Receive channels:
0000: 1 DMA Rx Channel
0001: 2 DMA Rx Channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL Transmit Queues
This field indicates the number of MTL Transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL Receive Queues
This field indicates the number of MTL Receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128

Reset value: 0x0C33 0031

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]			ESTSEL
		r	r	r	r					r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	r	r	r	r	r	r				r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ASP[1:0]**: Automotive Safety Package
Following are the encoding for the different Safety features
00: None
01: ECC only
10: AS_NPPE
11: AS_PPE

Bit 27 **TBSSEL**: Time-based scheduling Enable
This bit is set to 1 when the time-based scheduling feature is selected.

Bit 26 **FPESEL**: Frame Preemption Enable
This bit is set to 1 when the Enable Frame preemption feature is selected.

Bits 25:22 Reserved, must be kept at reset value.

Bits 21:20 **ESTWID[1:0]**: Width of the Time Interval field in the Gate Control List
This field indicates the width of the Configured Time Interval Field
00: No width
01: 16
10: 20
11: 24

Bits 19:17 **ESTDEP[2:0]**: Depth of the Gate Control List
This field indicates the depth of Gate Control list
000: No depth
001: 64
010: 128
011: 256
100: 512
101: 1024
110: Reserved, must not be used

Bit 16 **ESTSEL**: Enhancements to Scheduled Traffic Enable
This bit is set to 1 when the enable enhancements to scheduling traffic feature is selected.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **FRPES[1:0]**: Flexible Receive Parser Table Entries size
This field indicates the maximum number of parser entries supported by flexible receive parser.
00: 64 entries
01: 128 entries
10: 256 entries
11: Reserved, must not be used

Bits 12:11 **FRPBS[1:0]**: Flexible Receive Parser Buffer size
This field indicates the supported maximum number of bytes of the packet data to be parsed by flexible receive parser.
00: 64 bytes
01: 128 bytes
10: 256 bytes
11: Reserved, must not be used

Bit 10 **FRPSEL**: Flexible Receive Parser Selected
This bit is set to 1 when the Enable Flexible Programmable Receive Parser option is selected.

Bit 9 **PDUPSEL**: Broadcast/Multicast Packet Duplication
This bit is set to 1 when the Broadcast/Multicast Packet Duplication feature is selected.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
000: No Extended Rx VLAN Filters
001: 4 Extended Rx VLAN Filters
010: 8 Extended Rx VLAN Filters
011: 16 Extended Rx VLAN Filters
100: 24 Extended Rx VLAN Filters
101: 32 Extended Rx VLAN Filters
110 to 111: Reserved, must not be used

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble Suppression Enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back to Back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (GMII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical Layer Address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device Address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of Training Clocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR Clock Range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):
0000: MDC clock = eth_hclk / 42
0001: MDC clock = eth_hclk / 62
0010: MDC clock = eth_hclk / 16
0011: MDC clock = eth_hclk / 26
0100: MDC clock = eth_hclk / 102
0101: MDC clock = eth_hclk / 124
0110: MDC clock = eth_hclk / 204
0111: MDC clock = eth_hclk / 324
The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range.
When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks:
1000: eth_hclk / 4
1001: eth_hclk / 6
1010: eth_hclk / 8
1011: eth_hclk / 10
1100: eth_hclk / 12
1101: eth_hclk / 14
1110: eth_hclk / 16
1111: eth_hclk / 18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: GMII Operation Command

This bit indicates the operation command to the PHY.
00: Reserved, must not be used
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read
When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **GB**: GMII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in *MDIO address register (ETH_MACMDIOAR)* and *MDIO data register (ETH_MACMDIODR)* as long as this bit is set.

For write transfers, the application must first write 16-bit data in the GD field (and also RA field when C45E is set) in *MDIO data register (ETH_MACMDIODR)* register before setting this bit. When C45E is set, it should also write into the RA field of *MDIO data register (ETH_MACMDIODR)* before initiating a read transfer. When a read transfer is completed (GMII busy=0), the data read from the PHY register is valid in the GD field of the *MDIO data register (ETH_MACMDIODR)*.

Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in *MDIO address register (ETH_MACMDIOAR)* This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RA[15:0]**: Register Address

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **GD[15:0]**: GMII Data

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0210

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPPA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARPPA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230

Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SEEN**: Slave Error Response Enable

When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.

When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.

When this bit is reset, the access mode to these register fields remains rc_r (clear on read).

FPE control and status register (ETH_MACFPECSR)

Address offset: 0x0234

Reset value: 0x0000 0000

This register controls the operation of Frame Preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSP	TVER	RRSP	RVER
												rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARV	SRSP	SVER	EFPE
												rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TRSP**: Transmitted Respond Frame

Set when a Respond mPacket is transmitted (triggered by setting SRSP field). An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 18 **TVER**: Transmitted Verify Frame

Set when a Verify mPacket is transmitted (triggered by setting SVER field). An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 17 **RRSP**: Received Respond Frame

Set when a Respond mPacket is received. An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 16 **RVER**: Received Verify Frame

Set when a Verify mPacket is received. An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ARV**: Autogenerate Respond mPacket on receiving Verify mPacket

When set, this bit enables the function that autogenerates a Respond mPacket on receiving a Verify mPacket.

Bit 2 **SRSP**: Send Respond mPacket

When set, this bit indicates to the hardware that a Respond mPacket must be sent.
It is cleared by hardware after sending the Respond mPacket.
Writing 1 sets. Self-cleared.
Access restriction applies: this bit can be set by software or by hardware when the ARV bit is set and a verify mPacket is received. Setting it to 0 has no effect.

Bit 1 **SVER**: Send Verify mPacket

When set, this bit indicates to the hardware that a Verify mPacket must be sent. Reset by hardware after sending the Verify mPacket.
Writing 1 sets. Self-cleared.

Bit 0 **EFPE**: Enable Tx Frame Preemption

When set, Frame Preemption Tx functionality is enabled.

MAC presentation time register (ETH_MACPRSTIMR)

Address offset: 0x0240

Reset value: 0x0000 0000

This register contains the 32-bit binary rollover equivalent time of the PTP System Time expressed in ns.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTN[31:0]**: MAC 1722 Presentation Time in ns

These bits indicate the value of the 32-bit binary rollover equivalent time of the PTP System Time in ns.

MAC presentation time update register (ETH_MACPRSTIMUR)

Address offset: 0x0244

Reset value: 0x0000 0000

This register holds the 32-bit value of MAC 1722 Presentation Time in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control Register (ETH_MACTSCR)*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTU[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTU[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTU[31:0]**: MAC 1722 Presentation Time Update

This field holds the initialization value or the update value for the presentation time. When used for update, this field holds the 32-bit value, in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control Register (ETH_MACTSCR)*). When ADDSUB field of *System time nanoseconds update register (ETH_MACSTNUR)* is set, this value is directly used for subtraction.

MAC Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the GMII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is

received (0x11 in lane 0 of the first column) on the GMII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the GMII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

This bit is always set to 1.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **DCS**: DMA Channel Select

This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed.
0: DMA Rx channel 0
1: DMA Rx channel 1

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]

This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)

This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS
rw	rw	rw	rw	rw	rw	rw	rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address

When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.

0: DA

1: SA

Bits 29:24 **MBC[5:0]**: Mask Byte Control

These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

Bit 29: ETH_MACAxHR[15:8]

Bit 28: ETH_MACAxHR[7:0]

Bit 27: ETH_MACAxLR[31:24]

Bit 26: ETH_MACAxLR[23:16]

Bit 25: ETH_MACAxLR[15:8]

Bit 24: ETH_MACAxLR[7:0]

You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **DCS**: DMA Channel Select

This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed.

0: DMA Rx channel 0

1: DMA Rx channel 1

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]

This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover

When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Receive statistics counters.

The MMC receive interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, interrupts are set but the counter remains at all-ones. The MMC receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	RXCTRLPIS	RXRCVERRPIS	RXWDOGPIS	RXVLANGBPIS	RXFOVPIS	RXPAUSPIS	RXORANGEPIS	RXLENERPIS	RXUCGPIS	RX1024TMAXOCTGBPIS
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBPIS	RX256T511OCTGBPIS	RX128T255OCTGBPIS	RX65T127OCTGBPIS	RX64OCTGBPIS	RXOSIZEGPIS	RXUSIZEGPIS	RXJABERPIS	RXRUNTPIS	RXALGNERPIS	RXCRCERPIS	RXMCGPIS	RXBCGPIS	RXGOCTIS	RXGBOCTIS	RXGBPKTIS
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC Receive LPI transition counter interrupt status
This bit is set when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status
This bit is set when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 25 **RXCTRLPIS**: MMC Receive Control Packet Counter Interrupt Status
This bit is set when the *Rx control packets good register (ETH_RX_CONTROL_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 24 **RXRCVERRPIS**: MMC Receive Error Packet Counter Interrupt Status
This bit is set when the *Rx receive error register (ETH_RX_RECEIVE_ERROR)* counter reaches half of the maximum value or the maximum value.

Bit 23 **RXWDOGPIS**: MMC Receive Watchdog Error Packet Counter Interrupt Status
This bit is set when the *Rx watchdog error packets register (ETH_RX_WATCHDOG_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 22 **RXVLANGBPIS**: MMC Receive VLAN Good Bad Packet Counter Interrupt Status
This bit is set when the *Rx VLAN packets good bad register (ETH_RX_VLAN_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 21 **RXFOVPIS**: MMC Receive FIFO Overflow Packet Counter Interrupt Status
This bit is set when the *Rx FIFO overflow packets register (ETH_RX_FIFO_OVERFLOW_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 20 **RXPAUSPIS**: MMC Receive Pause Packet Counter Interrupt Status
This bit is set when the *Rx pause packets register (ETH_RX_PAUSE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 19 **RXORANGEPIS**: MMC Receive Out Of Range Error Packet Counter Interrupt Status
This bit is set when the *Rx out of range type packets register (ETH_RX_OUT_OF_RANGE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 18 **RXLENERPIS**: MMC Receive Length Error Packet Counter Interrupt Status
This bit is set when the *Rx length error packets register (ETH_RX_LENGTH_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status
This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 16 **RX1024TMAXOCTGBPIS**: MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status
This bit is set when the *Rx 1024 to max octets packets good bad register (ETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 15 **RX512T1023OCTGBPIS**: MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Rx 512 to 1023 octets packets good bad register (ETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 14 **RX256T511OCTGBPIS**: MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Rx 256 to 511 octets packets good bad register (ETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 13 **RX128T255OCTGBPIS**: MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Rx 128 to 255 octets packets good bad register (ETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 12 **RX65T127OCTGBPIS**: MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Rx 65 to 127 octets packets good bad register (ETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 11 **RX64OCTGBPIS**: MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 10 **RXOSIZEGPIS**: MMC Receive Oversize Good Packet Counter Interrupt Status

This bit is set when the *Rx oversize packets good register (ETH_RX_OVERSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 9 **RXUSIZEGPIS**: MMC Receive Undersize Good Packet Counter Interrupt Status

This bit is set when the *Rx undersize packets good register (ETH_RX_UNDERSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 8 **RXJABERPIS**: MMC Receive Jabber Error Packet Counter Interrupt Status

This bit is set when the *Rx jabber error packets register (ETH_RX_JABBER_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 7 **RXRUNTPIS**: MMC Receive Runt Packet Counter Interrupt Status

This bit is set when the *Rx runt error packets register (ETH_RX_RUNT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status

This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC Receive CRC Error Packet Counter Interrupt Status

This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 4 **RXMCGPIS**: MMC Receive Multicast Good Packet Counter Interrupt Status
This bit is set when the *Rx multicast packets good register (ETH_RX_MULTICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 3 **RXBCGPIS**: MMC Receive Broadcast Good Packet Counter Interrupt Status
This bit is set when the *Rx broadcast packets good register (ETH_RX_BROADCAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 2 **RXGOCTIS**: MMC Receive Good Octet Counter Interrupt Status
This bit is set when the *Rx octet count good register (ETH_RX_OCTET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 1 **RXGBOCTIS**: MMC Receive Good Bad Octet Counter Interrupt Status
This bit is set when the *Rx octet count good bad register (ETH_RX_OCTET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 0 **RXGBPKTIS**: MMC Receive Good Bad Packet Counter Interrupt Status
This bit is set when the *Rx packets count good bad register (ETH_RX_PACKETS_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	TXOSIZEGPIS	TXVLANGPIS	TXPAUSPIS	TXEXDEFPIS	TXGPKTIS	TXGOCTIS	TXCARERPIS	TXEXCOLPIS	TXLATCOLPIS	TXDEFPIS
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	TXUFLOWERRPIS	TXBCGBPIS	TXMCGBPIS	TXUCGBPIS	TX1024TMAXOCTGBPIS	TX512T1023OCTGBPIS	TX256T511OCTGBPIS	TX128T255OCTGBPIS	TX65T127OCTGBPIS	TX64OCTGBPIS	TXMCGPIS	TXBCGPIS	TXGBPKTIS	TXGBOCTIS
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC Transmit LPI transition counter interrupt status
This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status
This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 25 **TXOSIZEGPIS**: MMC Transmit Oversize Good Packet Counter Interrupt Status
This bit is set when the *Tx OSsize packets good register (ETH_TX_OSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 24 **TXVLANGPIS**: MMC Transmit VLAN Good Packet Counter Interrupt Status
This bit is set when the *Tx VLAN packets good register (ETH_TX_VLAN_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 23 **TXPAUSPIS**: MMC Transmit Pause Packet Counter Interrupt Status
This bit is set when the *Tx pause packets register (ETH_TX_PAUSE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 22 **TXEXDEFPIS**: MMC Transmit Excessive Deferral Packet Counter Interrupt Status
This bit is set when the *Tx excessive deferral error register (ETH_TX_EXCESSIVE_DEFERRAL_ERROR)* counter reaches half of the maximum value or the maximum value.

Bit 21 **TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status
This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 20 **TXGOCTIS**: MMC Transmit Good Octet Counter Interrupt Status
This bit is set when the *Tx octet count good register (ETH_TX_OCTET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 19 **TXCARERPIS**: MMC Transmit Carrier Error Packet Counter Interrupt Status
This bit is set when the *Tx carrier error packets register (ETH_TX_CARRIER_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 18 **TXEXCOLPIS**: MMC Transmit Excessive Collision Packet Counter Interrupt Status
This bit is set when the *Tx excessive collision packets register (ETH_TX_EXCESSIVE_COLLISION_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 17 **TXLATCOLPIS**: MMC Transmit Late Collision Packet Counter Interrupt Status
This bit is set when the *Tx late collision packets register (ETH_TX_LATE_COLLISION_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 16 **TXDEFPIS**: MMC Transmit Deferred Packet Counter Interrupt Status
This bit is set when the *Tx deferred packets register (ETH_TX_DEFERRED_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 15 **TXMCOLGPIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 13 **TXUFLOWERPIS**: MMC Transmit Underflow Error Packet Counter Interrupt Status
This bit is set when the *Tx underflow error packets register (ETH_TX_UNDERFLOW_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 12 **TXBCGBPIS**: MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status
This bit is set when the *Tx broadcast packets good bad register (ETH_TX_BROADCAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 11 **TXMCGBPIS**: MMC Transmit Multicast Good Bad Packet Counter Interrupt Status
This bit is set when the *Tx multicast packets good bad register (ETH_TX_MULTICAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 10 **TXUCGBPIS**: MMC Transmit Unicast Good Bad Packet Counter Interrupt Status
This bit is set when the *Tx unicast packets good bad register (ETH_TX_UNICAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 9 **TX1024TMAXOCTGBPIS**: MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status
This bit is set when the *Tx 1024 to max octets packets good bad register (ETH_TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 8 **TX512T1023OCTGBPIS**: MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status
This bit is set when the *Tx 512 to 1023 octets packets good bad register (ETH_TX_512TO1023OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 7 **TX256T511OCTGBPIS**: MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Tx 256 to 511 octets packets good bad register (ETH_TX_256TO511OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 6 **TX128T255OCTGBPIS**: MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Tx 128 to 255 octets packets good bad register (ETH_TX_128TO255OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 5 **TX65T127OCTGBPIS**: MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Tx 65 to 127 octets packets good bad register (ETH_TX_64TO127OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 4 **TX64OCTGBPIS**: MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status

This bit is set when the *Tx 64 octets packets good bad register (ETH_TX_64OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 3 **TXMCGPIS**: MMC Transmit Multicast Good Packet Counter Interrupt Status

This bit is set when the *Tx multicast packets good register (ETH_TX_MULTICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 2 **TXBCGPIS**: MMC Transmit Broadcast Good Packet Counter Interrupt Status

This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 1 **TXGBPKTIS**: MMC Transmit Good Bad Packet Counter Interrupt Status

This bit is set when the *Tx packet count good bad register (ETH_TX_PACKET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 0 **TXGBOCTIS**: MMC Transmit Good Bad Octet Counter Interrupt Status

This bit is set when the *Tx octet count good bad register (ETH_TX_OCTET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC receive interrupt mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	RXCTRLPIM	RXRCVERRPIM	RXWDOGPIM	RXVLANGBPIM	RXFOVPIM	RXPAUSPIM	RXORANGEPIM	RXLENERPIM	RXUCGPIM	RX1024TMAXOCTGBPIM
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBPIM	RX256T511OCTGBPIM	RX128T255OCTGBPIM	RX65T127OCTGBPIM	RX64OCTGBPIM	RXOSIZEGPIM	RXUSIZEGPIM	RXJABERPIM	RXRUNTPIM	RXALGNERPIM	RXCRCERPIM	RXMCGPIM	RXBCGPIM	RXGOCTIM	RXGBOCTIM	RXGBPKTIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 25 **RXCTRLPIM**: MMC Receive Control Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx control packets good register (ETH_RX_CONTROL_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 24 **RXRCVERRPIM**: MMC Receive Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx receive error register (ETH_RX_RECEIVE_ERROR)* counter reaches half of the maximum value or the maximum value.

Bit 23 **RXWDOGPIM**: MMC Receive Watchdog Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx watchdog error packets register (ETH_RX_WATCHDOG_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 22 **RXVLANGBPIM**: MMC Receive VLAN Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx VLAN packets good bad register (ETH_RX_VLAN_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 21 **RXFOVPIM**: MMC Receive FIFO Overflow Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx FIFO overflow packets register (ETH_RX_FIFO_OVERFLOW_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 20 **RXPAUSPIM**: MMC Receive Pause Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx pause packets register (ETH_RX_PAUSE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 19 **RXORANGEPIM**: MMC Receive Out Of Range Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx out of range type packets register (ETH_RX_OUT_OF_RANGE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 18 **RXLENERPIM**: MMC Receive Length Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx length error packets register (ETH_RX_LENGTH_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 17 **RXUCGPIM**: MMC Receive Unicast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 16 **RX1024TMAXOCTGBPIM**: MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx 1024 to max octets packets good bad register (ETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 15 **RX512T1023OCTGBPIM**: MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx 512 to 1023 octets packets good bad register (ETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 14 **RX256T511OCTGBPIM**: MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx 256 to 511 octets packets good bad register (ETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 13 **RX128T255OCTGBPIM**: MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx 128 to 255 octets packets good bad register (ETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 12 **RX65T127OCTGBPIM**: MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx 65 to 127 octets packets good bad register (ETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 11 **RX64OCTGBPIM**: MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 10 **RXOSIZEGPIM**: MMC Receive Oversize Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx oversize packets good register (ETH_RX_OVERSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 9 **RXUSIZEGPIM**: MMC Receive Undersize Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx undersize packets good register (ETH_RX_UNDERSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 8 **RXJABERPIM**: MMC Receive Jabber Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx jabber error packets register (ETH_RX_JABBER_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 7 **RXRUNTPIM**: MMC Receive Runt Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx runt error packets register (ETH_RX_RUNT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 6 **RXALGNERPIM**: MMC Receive Alignment Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 4 **RXMCGPIM**: MMC Receive Multicast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx multicast packets good register (ETH_RX_MULTICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 3 **RXBCGPIM**: MMC Receive Broadcast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx broadcast packets good register (ETH_RX_BROADCAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 2 **RXGOCTIM**: MMC Receive Good Octet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx octet count good register (ETH_RX_OCTET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 1 **RXGBOCTIM**: MMC Receive Good Bad Octet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx octet count good bad register (ETH_RX_OCTET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 0 **RXGBPKTIM**: MMC Receive Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Rx packets count good bad register (ETH_RX_PACKETS_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

The MMC transmit interrupt mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	TXOSIZEGPIM	TXVLANGPIM	TXPAUSPIM	TXEXDEFPIM	TXGPKTIM	TXGOCTIM	TXCARERPIM	TXEXCOLPIM	TXLATCOLPIM	TXDEFPIM
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	TXUFLOWERPIM	TXBCGBPIM	TXMCGBPIM	TXUCGBPIM	TX1024TMAXOCTGBPIM	TX512T1023OCTGBPIM	TX256T511OCTGBPIM	TX128T255OCTGBPIM	TX65T127OCTGBPIM	TX64OCTGBPIM	TXMCGPIM	TXBCGPIM	TXGBPKTIM	TXGBOCTIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC Transmit LPI transition counter interrupt Mask

Setting this bit masks the interrupt when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC Transmit LPI microsecond counter interrupt Mask

Setting this bit masks the interrupt when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 25 **TXOSIZEGPIM**: MMC Transmit Oversize Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx OSsize packets good register (ETH_TX_OSIZE_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 24 **TXVLANGPIM**: MMC Transmit VLAN Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx VLAN packets good register (ETH_TX_VLAN_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 23 **TXPAUSPIM**: MMC Transmit Pause Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx pause packets register (ETH_TX_PAUSE_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 22 **TXEXDEFPIM**: MMC Transmit Excessive Deferral Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx excessive deferral error register (ETH_TX_EXCESSIVE_DEFERRAL_ERROR)* counter reaches half of the maximum value or the maximum value.

Bit 21 **TXGPKTIM**: MMC Transmit Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 20 **TXGOCTIM**: MMC Transmit Good Octet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx octet count good register (ETH_TX_OCTET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 19 **TXCARERPIM**: MMC Transmit Carrier Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx carrier error packets register (ETH_TX_CARRIER_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 18 **TXEXCOLPIM**: MMC Transmit Excessive Collision Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx excessive collision packets register (ETH_TX_EXCESSIVE_COLLISION_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 17 **TXLATCOLPIM**: MMC Transmit Late Collision Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx late collision packets register (ETH_TX_LATE_COLLISION_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 16 **TXDEFPIM**: MMC Transmit Deferred Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx deferred packets register (ETH_TX_DEFERRED_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 15 **TXMCOLGPIM**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC Transmit Single Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 13 **TXUFLOWERPIM**: MMC Transmit Underflow Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx underflow error packets register (ETH_TX_UNDERFLOW_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 12 **TXBCGBPIM**: MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx broadcast packets good bad register (ETH_TX_BROADCAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 11 **TXMCGBPIM**: MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx multicast packets good bad register (ETH_TX_MULTICAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 10 **TXUCGBPIM**: MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx unicast packets good bad register (ETH_TX_UNICAST_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 9 **TX1024TMAXOCTGBPIM**: MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 1024 to max octets packets good bad register (ETH_TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 8 **TX512T1023OCTGBPIM**: MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 512 to 1023 octets packets good bad register (ETH_TX_512TO1023OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 7 **TX256T511OCTGBPIM**: MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 256 to 511 octets packets good bad register (ETH_TX_256TO511OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 6 **TX128T255OCTGBPIM**: MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 128 to 255 octets packets good bad register (ETH_TX_128TO255OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 5 **TX65T127OCTGBPIM**: MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 65 to 127 octets packets good bad register (ETH_TX_64TO127OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 4 **TX64OCTGBPIM**: MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx 64 octets packets good bad register (ETH_TX_64OCTETS_PACKETS_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 3 **TXMCGPIM**: MMC Transmit Multicast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx multicast packets good register (ETH_TX_MULTICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bit 2 **TXBCGPIM**: MMC Transmit Broadcast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 1 **TXGBPKTIM**: MMC Transmit Good Bad Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx packet count good bad register (ETH_TX_PACKET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Bit 0 **TXGBOCTIM**: MMC Transmit Good Bad Octet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx octet count good bad register (ETH_TX_OCTET_COUNT_GOOD_BAD)* counter reaches half of the maximum value or the maximum value.

Tx octet count good bad register (ETH_TX_OCTET_COUNT_GOOD_BAD)

Address offset: 0x0714

Reset value: 0x0000 0000

This register provides the number of bytes transmitted by the Ethernet peripheral, exclusive of preamble and retried bytes, in good and bad packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXOCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXOCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXOCTGB[31:0]**: Tx Octet Count Good Bad
This field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets.

Tx packet count good bad register (ETH_TX_PACKET_COUNT_GOOD_BAD)

Address offset: 0x0718

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral, exclusive of retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTGB[31:0]**: Tx Packet Count Good Bad
This field indicates the number of good and bad packets transmitted, exclusive of retried packets.

Tx broadcast packets good register (ETH_TX_BROADCAST_PACKETS_GOOD)

Address offset: 0x071C

Reset value: 0x0000 0000

This register provides the number of good broadcast packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXBCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXBCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXBCASTG[31:0]**: Tx Broadcast Packets Good

This field indicates the number of good broadcast packets transmitted.

Tx multicast packets good register (ETH_TX_MULTICAST_PACKETS_GOOD)

Address offset: 0x0720

Reset value: 0x0000 0000

This register provides the number of good multicast packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMCASTG[31:0]**: Tx multicast Packets Good

This field indicates the number of good multicast packets transmitted.

Tx 64 octets packets good bad register (ETH_TX_64OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x0724

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length of 64 bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX64OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX64OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX64OCTGB[31:0]**: Tx 64 octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets.

Tx 65 to 127 octets packets good bad register (ETH_TX_64TO127OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x0728

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX65_127OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX65_127OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX65_127OCTGB[31:0]**: Tx 65 to 127 octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

Tx 128 to 255 octets packets good bad register (ETH_TX_128TO255OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x072C

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX128_255OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX128_255OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX128_255OCTGB[31:0]**: Tx 128 to 255 octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

Tx 256 to 511 octets packets good bad register (ETH_TX_256TO511OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x0730

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX256_511OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX256_511OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX256_511OCTGB[31:0]**: Tx 256 to 511 octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

Tx 512 to 1023 octets packets good bad register (ETH_TX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x0734

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX512_1023OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX512_1023OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX512_1023OCTGB[31:0]**: Tx 512 to 1023 octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

Tx 1024 to max octets packets good bad register (ETH_TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Address offset: 0x0738

Reset value: 0x0000 0000

This register provides the number of good and bad packets transmitted by the Ethernet peripheral with a length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX1024_MAXOCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX1024_MAXOCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX1024_MAXOCTGB[31:0]**: Tx 1024 to Max octets Packets Good Bad

This field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets.

Tx unicast packets good bad register (ETH_TX_UNICAST_PACKETS_GOOD_BAD)

Address offset: 0x073C

Reset value: 0x0000 0000

This register provides the number of good and bad unicast packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXUCASTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXUCASTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXUCASTGB[31:0]**: Tx Unicast Packets Good Bad

This field indicates the number of good and bad unicast packets transmitted.

Tx multicast packets good bad register (ETH_TX_MULTICAST_PACKETS_GOOD_BAD)

Address offset: 0x0740

Reset value: 0x0000 0000

This register provides the number of good and bad multicast packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMCASTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCASTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMCASTGB[31:0]**: Tx Multicast Packets Good Bad

This field indicates the number of good and bad multicast packets transmitted.

Tx broadcast packets good bad register (ETH_TX_BROADCAST_PACKETS_GOOD_BAD)

Address offset: 0x0744

Reset value: 0x0000 0000

This register provides the number of good and bad broadcast packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXBCASTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXBCASTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXBCASTGB[31:0]**: Tx Broadcast Packets Good Bad

This field indicates the number of good and bad broadcast packets transmitted.

Tx underflow error packets register (ETH_TX_UNDERFLOW_ERROR_PACKETS)

Address offset: 0x0748

Reset value: 0x0000 0000

This register provides the number of packets aborted by the Ethernet peripheral because of packet underflow error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXUNDRFLW[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXUNDRFLW[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXUNDRFLW[31:0]**: Tx Underflow Error Packets

This field indicates the number of packets aborted because of packets underflow error.

Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets

This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets

This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.

Tx deferred packets register (ETH_TX_DEFERRED_PACKETS)

Address offset: 0x0754

Reset value: 0x0000 0000

This register provides the number of packets successfully transmitted by the Ethernet peripheral after a deferral in Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDEFRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDEFRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXDEFRD[31:0]**: Tx Deferred Packets

This field indicates the number of packets successfully transmitted after a deferral in the half-duplex mode.

Tx late collision packets register (ETH_TX_LATE_COLLISION_PACKETS)

Address offset: 0x0758

Reset value: 0x0000 0000

This register provides the number of packets aborted by the Ethernet peripheral because of late collision error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLATECOL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLATECOL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLATECOL[31:0]**: Tx Late Collision Packets

This field indicates the number of packets aborted because of late collision error.

Tx excessive collision packets register (ETH_TX_EXCESSIVE_COLLISION_PACKETS)

Address offset: 0x075C

Reset value: 0x0000 0000

This register provides the number of packets aborted by the Ethernet peripheral because of excessive (16) collision errors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXEXSCOL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXEXSCOL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXEXSCOL[31:0]**: Tx Excessive Collision Packets

This field indicates the number of packets aborted because of excessive (16) collision errors.

Tx carrier error packets register (ETH_TX_CARRIER_ERROR_PACKETS)

Address offset: 0x0760

Reset value: 0x0000 0000

This register provides the number of packets aborted by the Ethernet peripheral because of carrier sense error (no carrier or loss of carrier).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCARR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCARR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCARR[31:0]**: Tx Carrier Error Packets

This field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier).

Tx octet count good register (ETH_TX_OCTET_COUNT_GOOD)

Address offset: 0x0764

Reset value: 0x0000 0000

This register provides the number of bytes transmitted by the Ethernet peripheral, exclusive of preamble, only in good packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXOCTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXOCTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXOCTG[31:0]**: Tx Octet Count Good

This field indicates the number of bytes transmitted, exclusive of preamble, only in good packets.

Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good

This field indicates the number of good packets transmitted.

Tx excessive deferral error register (ETH_TX_EXCESSIVE_DEFERRAL_ERROR)

Address offset: 0x076C

Reset value: 0x0000 0000

This register provides the number of packets aborted by the Ethernet peripheral because of excessive deferral error (deferred for more than two max-sized packet times).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXEXSDEF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXEXSDEF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXEXSDEF[31:0]**: Tx Excessive Deferral Error

This field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times).

Tx pause packets register (ETH_TX_PAUSE_PACKETS)

Address offset: 0x0770

Reset value: 0x0000 0000

This register provides the number of good Pause packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPAUSE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPAUSE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPAUSE[31:0]**: Tx Pause Packets

This field indicates the number of good Pause packets transmitted.

Tx VLAN packets good register (ETH_TX_VLAN_PACKETS_GOOD)

Address offset: 0x0774

Reset value: 0x0000 0000

This register provides the number of good VLAN packets transmitted by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXVLANG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXVLANG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXVLANG[31:0]**: Tx VLAN Packets Good

This field provides the number of good VLAN packets transmitted.

Tx OSsize packets good register (ETH_TX_OSIZE_PACKETS_GOOD)

Address offset: 0x0778

Reset value: 0x0000 0000

This register provides the number of packets transmitted by the Ethernet peripheral without errors and with a length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the *Operating mode configuration register (ETH_MACCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXOSIZG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXOSIZG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXOSIZG[31:0]**: Tx OSize Packets Good

This field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the *Operating mode configuration register (ETH_MACCR)*).

Rx packets count good bad register (ETH_RX_PACKETS_COUNT_GOOD_BAD)

Address offset: 0x0780

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXPKTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXPKTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXPKTGB[31:0]**: Rx Packets Count Good Bad

This field indicates the number of good and bad packets received.

Rx octet count good bad register (ETH_RX_OCTET_COUNT_GOOD_BAD)

Address offset: 0x0784

Reset value: 0x0000 0000

This register provides the number of bytes received by the Ethernet peripheral, exclusive of preamble, in good and bad packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXOCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXOCTGB[31:0]**: Rx Octet Count Good Bad

This field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

Rx octet count good register (ETH_RX_OCTET_COUNT_GOOD)

Address offset: 0x0788

Reset value: 0x0000 0000

This register provides the number of bytes received by the Ethernet peripheral, exclusive of preamble, only in good packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXOCTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOCTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXOCTG[31:0]**: Rx Octet Count Good

This field indicates the number of bytes received, exclusive of preamble, only in good packets.

Rx broadcast packets good register (ETH_RX_BROADCAST_PACKETS_GOOD)

Address offset: 0x078C

Reset value: 0x0000 0000

This register provides the number of good broadcast packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXBCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXBCASTG[31:0]**: Rx Broadcast Packets Good

This field indicates the number of good broadcast packets received.

Rx multicast packets good register (ETH_RX_MULTICAST_PACKETS_GOOD)

Address offset: 0x0790

Reset value: 0x0000 0000

This register provides the number of good multicast packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXMCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXMCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXMCASTG[31:0]**: Rx Multicast Packets Good

This field indicates the number of good multicast packets received.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets

This field indicates the number of packets received with CRC error.

Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets

This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx runt error packets register (ETH_RX_RUNT_ERROR_PACKETS)

Address offset: 0x079C

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with runt error (length less than 64 bytes and CRC error).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXRUNTERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXRUNTERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXRUNTERR[31:0]**: Rx Runt Error Packets

This field indicates the number of packets received with runt error (length less than 64 bytes and CRC error).

Rx jabber error packets register (ETH_RX_JABBER_ERROR_PACKETS)

Address offset: 0x07A0

Reset value: 0x0000 0000

This register provides the number of giant packets received by the Ethernet peripheral with a length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged), and with CRC error. If Jumbo Packet mode is enabled, the packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXJABERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXJABERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXJABERR[31:0]**: Rx Jabber Error Packets

This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

Rx undersize packets good register (ETH_RX_UNDERSIZE_PACKETS_GOOD)

Address offset: 0x07A4

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with a length less than 64 bytes, without any errors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUNDERSZG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUNDERSZG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUNDERSZG[31:0]**: Rx Undersize Packets Good

This field indicates the number of packets received with length less than 64 bytes, without any errors.

Rx oversize packets good register (ETH_RX_OVERSIZE_PACKETS_GOOD)

Address offset: 0x07A8

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral without errors, with a length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the *Operating mode configuration register (ETH_MACCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXOVERSZG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOVERSZG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXOVERSZG[31:0]**: Rx Oversize Packets Good

This field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the *Operating mode configuration register (ETH_MACCR)*).

Rx 64 octets packets good bad register (ETH_RX_64OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07AC

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with length 64 bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX64OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX64OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RX64OCTGB[31:0]**: Rx 64 Octets Packets Good Bad

This field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble.

Rx 65 to 127 octets packets good bad register (ETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07B0

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with a length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX65_127OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX65_127OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TX65_127OCTGB[31:0]**: Rx 65-127 Octets Packets Good Bad

This field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

Rx 128 to 255 octets packets good bad register (ETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07B4

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with a length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX128_255OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX128_255OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RX128_255OCTGB[31:0]**: Rx 128-255 Octets Packets Good Bad

This field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

Rx 256 to 511 octets packets good bad register (ETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07B8

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with a length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX256_511OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX256_511OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RX256_511OCTGB[31:0]**: Rx 256-511 Octets Packets Good Bad

This field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

Rx 512 to 1023 octets packets good bad register (ETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07BC

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with a length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX512_1023OCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512_1023OCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RX512_1023OCTGB[31:0]**: Rx 512-1023 Octets Packets Good Bad

This field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

Rx 1024 to max octets packets good bad register (ETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Address offset: 0x07C0

Reset value: 0x0000 0000

This register provides the number of good and bad packets received by the Ethernet peripheral with a length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX1024_MAXOCTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX1024_MAXOCTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RX1024_MAXOCTGB[31:0]**: Rx 1024-Max Octets Good Bad

This field indicates the number of good and bad packets received with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good

This field indicates the number of good unicast packets received.

Rx length error packets register (ETH_RX_LENGTH_ERROR_PACKETS)

Address offset: 0x07C8

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with a length error (Length Type field not equal to packet size), for all packets with valid length field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLENERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLENERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLENERR[31:0]**: Rx Length Error Packets

This field indicates the number of packets received with length error (Length Type field not equal to packet size), for all packets with valid length field.

Rx out of range type packets register (ETH_RX_OUT_OF_RANGE_PACKETS)

Address offset: 0x07CC

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with la ength field not equal to the valid packet size (greater than 1,500 but less than 1,536).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXOUTOFRNG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOUTOFRNG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXOUTOFRNG[31:0]**: Rx Out of Range Type Packet

This field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

Rx pause packets register (ETH_RX_PAUSE_PACKETS)

Address offset: 0x07D0

Reset value: 0x0000 0000

This register provides the number of good and valid Pause packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXPAUSEPKT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXPAUSEPKT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXPAUSEPKT[31:0]**: Rx Pause Packets
This field indicates the number of good and valid Pause packets received.

Rx FIFO overflow packets register (ETH_RX_FIFO_OVERFLOW_PACKETS)

Address offset: 0x07D4

Reset value: 0x0000 0000

This register provides the number of missed received packets because of FIFO overflow in the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFIFOOVFL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOOVFL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXFIFOOVFL[31:0]**: Rx FIFO Overflow Packets
This field indicates the number of missed received packetsbecause of FIFO overflow.

Rx VLAN packets good bad register (ETH_RX_VLAN_PACKETS_GOOD_BAD)

Address offset: 0x07D8

Reset value: 0x0000 0000

This register provides the number of good and bad VLAN packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXVLANPKTGB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXVLANPKTGB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXVLANPKTGB[31:0]**: Rx VLAN Packets Good Bad

This field indicates the number of good and bad VLAN packets received.

Rx watchdog error packets register (ETH_RX_WATCHDOG_ERROR_PACKETS)

Address offset: 0x07DC

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in *Operating mode configuration register (ETH_MACCR)*), 10,240 bytes (when JE bit is set and WD bit is reset in *Operating mode configuration register (ETH_MACCR)*), 16,384 bytes (when WD bit is set in *Operating mode configuration register (ETH_MACCR)*) or the value programmed in the watchdog timeout register).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXWDGERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWDGERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXWDGERR[31:0]**: Rx Watchdog Error Packets

This field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in *Operating mode configuration register (ETH_MACCR)*), 10,240 bytes (when JE bit is set and WD bit is reset in *Operating mode configuration register (ETH_MACCR)*), 16,384bytes (when WD bit is set in MAC_Configuration register) or the value programmed in the)*Watchdog and jabber timeout register (ETH_MACWJBTR)*.

Rx receive error register (ETH_RX_RECEIVE_ERROR)

Address offset: 0x07E0

Reset value: 0x0000 0000

This register provides the number of packets received by the Ethernet peripheral with Receive error or Packet Extension error on the GMII or MII interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXRCVERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXRCVERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXRCVERR[31:0]**: Rx Receive Error Packets

This field indicates the number of packets received with Receive error or Packet Extension error on the GMII or MII interface.

Rx control packets good register (ETH_RX_CONTROL_PACKETS_GOOD)

Address offset: 0x07E4

Reset value: 0x0000 0000

This register provides the number of good control packets received by the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCTRLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCTRLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCTRLG[31:0]**: Rx Control Packets Good

This field indicates the number of good control packets received.

Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the *LPI control and status register (ETH_MACLCSR)*), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter

This field indicates the number of times Rx LPI Entry has occurred.

MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)

Address offset: 0x08A0

Reset value: 0x0000 0000

This register maintains the interrupts generated from all FPE related Transmit statistics counters when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When CNTSTOPRO bit is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones. The MMC FPE transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIS	FCIS
														rc_r	rc_r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HRCIS**: MMC Tx Hold Request Counter Interrupt Status
This bit is set when the *MMC Tx hold request counter register (ETH_MMC_TX_HRCR)* reaches half of the maximum value or the maximum value.

Bit 0 **FCIS**: MMC Tx FPE Fragment Counter Interrupt status
This bit is set when the *MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)* reaches half of the maximum value or the maximum value.

MMC FPE Tx interrupt mask register (ETH_MMC_FPE_TX_IMR)

Address offset: 0x08A4

Reset value: 0x0000 0000

This register maintains the masks for interrupts generated from all FPE related Transmit statistics counters when FPE related receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIM	FCIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HRCIM**: MMC transmit Hold Request Counter Interrupt Mask

Setting this bit masks the interrupt when the *MMC Tx hold request counter register (ETH_MMC_TX_HRCR)* reaches half of the maximum value or the maximum value.

Bit 0 **FCIM**: MMC Transmit Fragment Counter Interrupt Mask

Setting this bit masks the interrupt when the *MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)* reaches half of the maximum value or the maximum value.

MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)

Address offset: 0x08A8

Reset value: 0x0000 0000

This register provides the number of additional mPackets transmitted due to preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFFC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXFFC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXFFC[31:0]**: Tx FPE Fragment counter

This field indicates the number of additional mPackets that has been transmitted due to preemption

MMC Tx hold request counter register (ETH_MMC_TX_HRCR)

Address offset: 0x08AC

Reset value: 0x0000 0000

This register provides the count of number of times a hold request is issue to the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXHRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXHRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXHRC[31:0]**: Tx hold request counter

This field indicates count of number of a hold requests issued to the MAC.

MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)

Address offset: 0x08C0

Reset value: 0x0000 0000

This register maintains the interrupts generated from all FPE related Receive statistics counters when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When CNTSTOPRO bit is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones. The MMC FPE receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIS	PAOCIS	PSECIS	PAECIS
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FCIS**: MMC Rx FPE Fragment Counter Interrupt Status

This bit is set when the *MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)* counter reaches half of the maximum value or the maximum value.

Bit 2 **PAOCIS**: MMC Rx Packet Assembly OK counter interrupt status
This bit is set when the *MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)* counter reaches half of the maximum value or the maximum value.

Bit 1 **PSECIS**: MMC Rx Packet SMD Error counter interrupt status
This bit is set when the *MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)* counter reaches half of the maximum value or the maximum value.

Bit 0 **PAECIS**: MMC Rx Packet Assembly Error counter interrupt status
This bit is set when the *MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)* counter reaches half of the maximum value or the maximum value.

MMC FPE Rx interrupt mask register (ETH_MMC_FPE_RX_IMR)

Address offset: 0x08C4

Reset value: 0x0000 0000

This register maintains the masks for interrupts generated from all FPE related Receive statistics counters when FPE related receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIM	PAOCIM	PSECIM	PAECIM
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FCIM**: MMC Rx FPE Fragment Counter Interrupt Mask
Setting this bit masks the interrupt when the *MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)* counter reaches half of the maximum value or the maximum value.

Bit 2 **PAOCIM**: MMC Rx packet Assembly OK Counter Interrupt Mask
Setting this bit masks the interrupt when the *MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)* counter reaches half of the maximum value or the maximum value.

Bit 1 **PSECIM**: MMC Rx Packet SMD Error Counter Interrupt Mask
Setting this bit masks the interrupt when the *MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)* counter reaches half of the maximum value or the maximum value.

Bit 0 **PAECIM**: MMC Rx Packet Assembly Error Counter Interrupt Mask
Setting this bit masks the interrupt when the *MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)* counter reaches half of the maximum value or the maximum value.

MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)

Address offset: 0x08C8

Reset value: 0x0000 0000

This register provides the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAEC[31:0]**: Rx Packet Assembly Error Counter

This field indicates the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value.

MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)

Address offset: 0x08CC

Reset value: 0x0000 0000

This register provides the number of received MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PSEC[31:0]**: Rx Packet SMD Error Counter

This field indicates the number of MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame

MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)

Address offset: 0x08D0

Reset value: 0x0000 0000

This register provides the number of MAC frames that were successfully reassembled and delivered to MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAOC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAOC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAOC[31:0]**: Rx Packet Assembly OK Counter

This field indicates the number of MAC frames that were successfully reassembled and delivered to MAC.

MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)

Address offset: 0x08D4

Reset value: 0x0000 0000

This register provides the number of additional mPackets received due to preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FFC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **FFC[31:0]**: Rx FPE Fragment Counter

This field indicates the number of additional mPackets received due to preemption

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DMCHEN0	Res.	Res.	Res.	DMCHN0	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
			rw				rw			rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **DMCHEN0**: DMA Channel Select Enable

When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DMCHN0**: DMA Channel Number

When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. The width of this field depends on the number of the DMA channels present in your configuration.

0: DMA channel 0

1: DMA channel 1

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 Address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 Address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 Address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 Address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

Layer3 Address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DMCHEN1	Res.	Res.	Res.	DMCHN1	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
			rw				rw			rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **DMCHEN1**: DMA Channel Select Enable

When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DMCHN1**: DMA Channel Number

When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. The width of this field depends on the number of the DMA channels present in your configuration.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4SPM1 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable
When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.
The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match
Condition: IPv4 packets
This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:
0: No bits are masked.
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
..
31: All bits except MSb are masked.
Condition: IPv6 packets
Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:
0: No bits are masked
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
..
127: All bits except MSb are masked
This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match
Condition: IPv4 packets
This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:
0: No bits are masked.
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
..
31: All bits except MSb are masked.

Condition: IPv6 packets
This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN1 and L3SAM1bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

MAC Indirect Access Control register (ETH_MAC_IACR)

Address offset: 0x0A70

Reset value: 0x0000 0000

This register provides the Indirect Access control and status for the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSEL[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AOFF[7:0]								Res.	Res.	AUTO	Res.	Res.	Res.	COM	OB
rw	rw	rw	rw	rw	rw	rw	rw			rw				rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **MSEL[3:0]**: Mode Select

This field is used for indirect accesses to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.

0000:Typ_RXQ_ (Type-based RXQ mapping)

0001-1111: Reserved for future use

Bits 15:8 **AOFF[7:0]**: Address Offset

This field is used for indirect accesses to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.

00000000: IndReg0 (Indirect register 0)

00000001: IndReg1 (Indirect register 1)

...

00000111: IndReg7 (Indirect register 7)

00001000-11111111: Reserved, must not be used

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **AUTO**: Auto-increment

1: AOFF is incremented by 1. The software should ensure not to cause a wrap condition. Byte-wise read/write is not supported when auto-increment is enabled.

0: AOFF is not automatically incremented. The software should program the correct address offset for each access.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **COM**: Command type

This bit indicates the register access type:
1: Indicates a read operation.
0: Indicates a write operation.

Bit 0 **OB**: Operation Busy.

This bit is set along with a read or write command to initiate an indirect access to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)*. This bit is reset when the read or write command to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* is complete. The next indirect register access can be initiated only after this bit is reset. During a write operation, the bit is reset only after the data has been written to the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* register. During a read operation, the data should be read from the *MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)* only after this bit is reset.

MAC type-based Rx Queue mapping register (ETH_MAC_TMRQR)

Address offset: 0x0A74

Reset value: 0x0000 0000

This register contains the type, associated queue number and packet type (Preemption/Express) related to Type-based RXQ mappings. It holds the data for eight indirect registers, defined by the AOFF[7:0] bitfield in the *MAC Indirect Access Control register (ETH_MAC_IACR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PFEX	Res.	TMRQ[2:0]		
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **PFEX**: Preemption or Express Packet

This bit indicates if it is a preemption packet or express packet.

0: Express packet

1: Preemption packet

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **TMRQ[2:0]**: Type Match Rx Queue Number

Indicates the receive queue number to which the packet needs to be forwarded on detecting a type match.

Bits 15:0 **TYP[15:0]**: Type field Value

Indicates the type value of packet that needs to be compared with the received Ethernet packet. This field is valid when programmed to a value greater than or equal to 0x0600.

Timestamp control Register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	ESTI	Res.	TSENMACADDR	SNAPTYPSEL[1:0]	
			rw				rw				rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **AV8021ASMEN**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When the PTP offload feature is enabled, for PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ESTI**: External System Time Input

When this bit is set, the MAC uses the external 64-bit reference System Time input for the following:

- To take the timestamp provided as status
- To insert the timestamp in transmit PTP packets when One-step Timestamp or Timestamp Offload feature is enabled.

When this bit is reset, the MAC uses the internal reference System Time.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in *Table 823: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see *Table 823: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PTGE**: Presentation Time Generation Enable

When this bit is set the Presentation Time generation is enabled.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.

When Media Clock Generation and Recovery is enabled, ETH_MACPRSTIMUR should also be updated before setting this field.

This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.

When Media Clock Generation and Recovery is enabled, *MAC presentation time update register (ETH_MACPRSTIMUR)* should also be updated before setting this field.

This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.

On the Receive side, the MAC processes the 1588 packets only if this bit is set.

Subsecond increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

This register specifies the value to be added to the internal system time register every cycle of clk_ptp_ref_i clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control Register (ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in *Timestamp control Register (ETH_MACTSCR)*, each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in *Timestamp control Register (ETH_MACTSCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.
When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.
For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} – 2$).

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in *System time nanoseconds update register (ETH_MACSTNUR)*. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, $2^{32} – 1$) and the *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD SUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field is the subseconds part of the update.

– ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.

– ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the *Timestamp control Register (ETH_MACTSCR)*.

– TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)*is 1:
- The programmed value must be 10^9 – <subsecond value>.
- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.

– TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)* is 0:
- The programmed value must be 2^{31} - <subsecond_value>.
- Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, 2^{31} – 1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset and 0x3B9A C9FF (that is, 10^9 –1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, 2^{32} – 1).

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the *Timestamp control Register (ETH_MACTSCR)*). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	rc_r					rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR1	TSTARGT1	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
rc_r										rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status Interrupt Status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **TSTRGTERR1**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTS1R and ETH_MACPPSTTSN1R registers elapses. It is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **TSTARGT1**: Timestamp Target Time Reached

When set, this bit indicates that the value of the system time is greater than or equal to the value specified in the ETH_MACPPSTTS1R and ETH_MACPPSTTSN1R registers.
This bit is cleared when the application reads this bit (or write it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTS0R and ETH_MACPPSTTSN0R elapses (see *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*). This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTS0R and ETH_MACPPSTTSN0R registers (see *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).
This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Rx domain time increment register (ETH_MACRXDTI)

Address offset: 0x0B24

Reset value: 0x0000 0000

This register indicates the increment value of the PTP time maintained in the Receive clock domain that is used for timestamping.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXNS[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **RXNS[15:0]**: Receive domain time increment value in nanoseconds

This bitfield indicates the nanosecond part of the increment value of the PTP time maintained in the Receive clock domain that is used for timestamping.

For example, when operating in 2.5G speed, the clock period is 3.2 ns, so in digital rollover mode, the RXNS[15:0] bitfield must be programmed to 3. In binary rollover mode, 3 / 0.466 = 6.44, so the RXNS[15:0] bitfield must be programmed to 6. When RXNS[15:0] is zero, the fixed value based on the operating speed is used for PTP time increment in Receive clock domain.

Bits 15:0 Reserved, must be kept at reset value.

Tx domain time increment register (ETH_MACTXDTI)

Address offset: 0x0B28

Reset value: 0x0000 0000

This register indicates the increment value of the PTP time maintained in the Transmit clock domain that is used for timestamping.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXNS[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **TXNS[15:0]**: Transmit domain time increment value in nanoseconds

This bitfield indicates the nanosecond part of the increment value of the PTP time maintained in the Transmit clock domain that is used for timestamping.

For example, when operating in 2.5G speed, the clock period is 3.2 ns, so in digital rollover mode, the TXNS[15:0] bitfield must be programmed to 3. In binary rollover mode, 3 / 0.466 = 6.44, so the TXNS[15:0] bitfield must be programmed to 6. When TXNS[15:0] is zero, the fixed value based on operating speed is used for PTP time increment in Transmit clock domain.

Bits 15:0 Reserved, must be kept at reset value.

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.The *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register, along with *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* is read

(bits[31:24]).
If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)*. The status bit TXTSSIS bit [15] in *Timestamp status register (ETH_MACTSSR)* register is set when the MAC transmitter captures the timestamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

– The timestamp of the current packet is ignored if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is reset

– The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp
Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp
Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction

This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000.

The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction

This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}.

For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD 8000, which is the 2's complement of 0x0002 8000(2.5×2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3×2^{16}).

Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Timestamp Ingress Latency register (ETH_MACTSILR)

Address offset: 0x0B68

Reset value: 0x0000 0000

This register holds the ingress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ITLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ITLNS[11:0]**: Ingress Timestamp Latency, in nanoseconds

This register holds the average latency (in nanoseconds) between the MAC input ports and the actual point (GMII/MII) where the ingress timestamp is taken. Ingress correction value is computed as described in *Section : Ingress correction*.

Bits 15:8 **ITLSNS[7:0]**: Ingress Timestamp Latency, in subnanoseconds

This register holds the average latency (in subnanoseconds) between the MAC input ports and the actual point (GMII/MII) where the ingress timestamp is taken. ingress correction value is computed as described in *Section : Ingress correction*.

Bits 7:0 Reserved, must be kept at reset value.

Timestamp Egress Latency register (ETH_MACTSELR)

Address offset: 0x0B6C

Reset value: 0x0000 0000

This register holds the Egress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ETLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ETLNS[11:0]**: Egress Timestamp Latency, in nanoseconds

This register holds the average latency in nanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 15:8 **ETLSNS[7:0]**: Egress Timestamp Latency, in subnanoseconds

This register holds the average latency in subnanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 7:0 Reserved, must be kept at reset value.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS0 Output

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.

0: PPS mode

1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCGREN1	TRGTMODSEL1[1:0]		Res.	PPSCMD1[3:0]				MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:16 Reserved, must be kept at reset value.

Bit 15 **MCGREN1**: MCGR Mode Enable for PPS Output 1

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set, it operates in MCGR mode and when reset it operates in PPS mode.
0: PPS mode
1: MCGR mode

Bits 14:13 **TRGTMODSEL1[1:0]**: Target Time Register Mode for PPS Output 1

This field indicates the target time register (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:
00: Target time registers are programmed only for generating the interrupt event.
01: Enabled MCGR Interrupt whose status bit is indicated by TSTARGT1 bit in *Timestamp status register (ETH_MACTSSR)* register
10: Target time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **PPSCMD1[3:0]:** Flexible PPS Output 1 Control

This field controls the flexible PPS output 1 signal. This field is similar to the PPSCMD[3:0] bitfield.

– When MCGREN1 is set, PPSCMD1 indicated by the four bits [11:8] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 1. This field is similar to the PPSCMD0 Presentation Time Control bits.
– When MCGREN1 is cleared, only the three bits [10:8] are used as PPSCMD1 and the 4th bit must be cleared.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS Output 0

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.

0: PPS mode

1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output 0

This field indicates the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output 0 signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output 0 signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output 0 Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output 0 (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are ‘all zero’. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse.

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.

0010: START Pulse Train.

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START.

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse Train at time.

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately.

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).

0110: Cancel STOP Pulse train.

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111 to 1111: Reserved, must not be used

PPS x target time seconds register (ETH_MACPPSTTSxR)

Address offset: 0x0B80 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS output x Target Time Seconds register, along with *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*, is used to schedule an interrupt event (Bit TSSOVF of *Timestamp status register (ETH_MACTSSR)*) when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the *PPS control register (ETH_MACPPSCR)*.

If the PTGE field of *Timestamp control Register (ETH_MACTSCR)* is set with Presentation time control set in Recovery mode, then these bits indicate the TPT being programmed by the application and in Generation mode it indicates the CPT generated at the sampled trigger.

PPS x target time nanoseconds register (ETH_MACPPSTTNxR)

Address offset: 0x0B84 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the *PPS control register (ETH_MACPPSCR)* is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in *PPS control register (ETH_MACPPSCR)*.

When the TSCTRLSSR bit is reset in the *Timestamp control Register (ETH_MACTSCR)*, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of subsecond increment value.

When the TSCTRLSSR bit is set in the *Timestamp control Register (ETH_MACTSCR)*, this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

PPS x interval register (ETH_MACPPSIxR)

Address offset: 0x0B88 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

PPS x width register (ETH_MACPPSWxR)

Address offset: 0x0B8C + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in PPS x interval register (ETH_MACPPSIxR).

PTP Offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								PDRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number

This field indicates the domain Number in which the PTP node is operating.

Bit 7 **PDRDIS**: Disable Peer Delay Response response generation

When this bit is set, the Peer Delay Response (Pdelay_Resp) response is not generated for received Peer Delay Request (Pdelay_Req) request packet, as required by the programmed mode.

Setting this bit to 1 affects the normal PTP Offload operation and the time synchronization. So, this bit must be set only if there is problem with Pdelay_Resp generation in hardware and/or Pdelay_Resp generation is handled by software.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response are not generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP Source Port Identity 0 Register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2
This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval

This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:

Delay_Req to SYNC Ratio

In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.

0: DelayReq generated for every received SYNC

1: DelayReq generated every alternate reception of SYNC

2: for every 4 SYNC messages

3: for every 8 SYNC messages

4: for every 16 SYNC messages

5: for every 32 SYNC messages

Others: Reserved, must not be used

The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:

Log Sync Interval

This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 888. Ethernet MAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE	PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC	Res.	Res.	GPSL[13:0]														
	Reset value		0	0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE	Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWJBTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PJE	Res.	Res.	Res.	Res.	JTO[3:0]					Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
	Reset value								0					0	0	0	0								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTCR	EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFS[1:0]		CT	OB
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0												0	0	0	0
0x0054	ETH_MACVTDR	Res.	Res.	Res.	Res.	Res.	Res.	DMACHN	DMACHEN	Res.	Res.	Res.	ERIVLT	ERSVLM	DOVLTC	ETV	VEN	VID[15:0]															
	Reset value							0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	ETH_MACVHTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	BUSY	RDWR	Res.	Res.	Res.	Res.	ADDR1	ADDR0	Res.	Res.	CBTI	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	ETH_MACVIR [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSV	Res.	Res.	Res.	VLT[15:0]															
	Reset value													0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0064	ETH_MACiVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																
0x0070	ETH_ MACQ0TXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094	ETH_MACRXQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE	Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
	Reset value															0	0							0	0							0	0
0x0098- 0x009C	Reserved																																
0x00A0	ETH_ MACRXQC0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
	Reset value																													0	0	0	0
0x00A4	ETH_ MACRXQC1R	Res.	Res.	TBRQE	OMCBCQ	Res.	FPRQ[2:0]			TPQC[1]	TPQC[0]	TACPQE	MCBCQEN	Res.	MCBCQ[2:0]			Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ[2:0]		
	Reset value			0	0		0	0	0	0	0	0	0		0	0	0		0	0	0						0	0	0		0	0	0
0x00A8	ETH_ MACRXQC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSRQ1[7:0]								PSRQ0[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFRIS	MFTIS	MDIOIS	FPEIS	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIII
	Reset value												0	0	0	0			0	0	0		0	0	0			0	0	0			0
0x00B4	ETH_MACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	FPEIE	Res.	Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	RGSMIIIE
	Reset value														0	0			0	0	0							0	0	0			0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00B8	ETH_ MACRXTXSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	0
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRS	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	MACRWKPFR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																
0x00D0	ETH_MACLCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN	Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value											0	0	0	0	0	0							0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]										TWT[15:0]															
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	ETH_MACLETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	ETH_ MAC1USTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
	Reset value																					0	0	0	0	0	1	1	0	0	0	1	1
0x00E0 - 0x00F4	Reserved																																
0x00F8	ETH_ MACPHYCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNKSTS	LNKSPEED[1:0]		LNKMOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUD	TC
	Reset value													0	0	0	0															0	0
0x00FC - 0x010C	Reserved																																
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]							SNPSVER[7:0]								
	Reset value	0x0000 1B53																															
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0															0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0118	Reserved																																
0x011C	ETH_MACHWF0R	ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]			MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]				Res.	RXCOESEL	Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	Reset value	0x0E0D 73F7																															
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]			Res.		HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]			SPRAM	RXFIFOSIZE[4:0]						
	Reset value	0x1114 59A5																															
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]		RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
	Reset value	0x420C 10C1																															
0x0128	ETH_MACHWF3R	Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]		ESTSEL	Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]			
	Reset value	0x0C33 0031																															
0x012C - 0x01FC	Reserved																																
0x0200	ETH_ MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0
0x0204	ETH_ MACMDIODR	RA[15:0]																GD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0210	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x022C	Reserved																																
0x0230	ETH_ MACCSRSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
	Reset value																								0								0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0234	ETH_MACFPECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSP	TVER	RRSP	RVER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARV	SRSP	SVER	EFPE
	Reset value													0	0	0	0													0	0	0	0
0x0238-0x23C	Reserved																																
0x0240	ETH_MACPRSTIMR	MPTN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0244	ETH_MACPRSTIMUR	MPTU[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0248 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	1															0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0700	ETH_MMC_ CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	ETH_MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	RXCTRLPIS	RXRCVERRPIS	RXWDOGPIS	RXVLANGBPIS	RXFOVPIS	RXPAUSPIS	RXORANGEPIS	RXLENERPIS	RXUCGPIS	RX1024TMAXOCTGBPIS	RX512T1023OCTGBPIS	RX256T511OCTGBPIS	RX128T255OCTGBPIS	RX65T127OCTGBPIS	RX64OCTGBPIS	RXOSIZEGPIS	RXUSIZEGPIS	RXJABERPIS	RXRUNTPIS	RXALGNERPIS	RXCRCERPIS	RXMCGPIS	RXBCGPIS	RXGOCTIS	RXGBOCTIS	RXGBPKTIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0708	ETH_MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	TXOSIZEGPIS	TXVLANGPIS	TXPAUSPIS	TXEXDEFPIS	TXGPKTIS	TXGOCTIS	TXCARERPIS	TXEXCOLPIS	TXLATCOLPIS	TXDEFPIS	TXMCOLGPIS	TXSCOLGPIS	TXUFLOWERPIS	TXBCGPIS	TXMCGBPIS	TXUCGBPIS	TX1024TMAXOCTGBPIS	TX512T1023OCTGBPIS	TX256T511OCTGBPIS	TX128T255OCTGBPIS	TX65T127OCTGBPIS	TX64OCTGBPIS	TXMCGPIS	TXBCGPIS	TXGBPKTIS	TXGBOCTIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070C	ETH_MMC_RX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	RXCTRLPIM	RXRCVERRPIM	RXWDOGPIM	RXVLANGBPIM	RXFOVPIM	RXPAUSPIM	RXORANGEPIM	RXLENERPIM	RXUCGPIM	RX1024TMAXOCTGBPIM.	RX512T1023OCTGBPIM	RX256T511OCTGBPIM	RX128T255OCTGBPIM	RX65T127OCTGBPIM	RX64OCTGBPIM	RXOSIZEGPIM	RXUSIZEGPIM	RXJABERPIM	RXRUNTPIM	RXALGNERPIM	RXCRCERPIM	RXMCGPIM	RXBCGPIM	RXGOCTIM	RXGBOCTIM	RXGBPKTIM
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0710	ETH_MMC_TX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	TXOSIZEGPIM	TXVLANGPIM	TXPAUSPIM	TXEXDEFPIM	TXGPKTIM	TXGOCTIM	TXCARERPIM	TXEXCOLPIM	TXLATCOLPIM	TXDEFPIM	TXMCOLGPIM	TXSCOLGPIM	TXUFLOWERPIM	TXBCGBPIM	TXMCGBPIM	TXUCGBPIS	TX1024TMAXOCTGBPIM	TX512T1023OCTGBPIM	TX256T511OCTGBPIM	TX128T255OCTGBPIM.	TX65T127OCTGBPIM	TX64OCTGBPIM	TXMCGPIM	TXBCGPIM	TXGBPKTIM	TXGBOCTIM
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0714	ETH_TX_OCTET_C OUNT_GOOD_BAD	TXOCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0718	ETH_TX_PACKET_ COUNT_GOOD_BA D	TXPKTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x071C	ETH_TX_BROADCAST_PACKETS_GOOD	TXBCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0720	ETH_TX_MULTICAST_PACKETS_GOOD	TXMCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0724	ETH_TX_64OCTETS_PACKETS_GOOD_BAD	TX64OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0728	ETH_TX_64TO127OCTETS_PACKETS_GOOD_BAD	TX65_127OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x072C	ETH_TX_128TO255OCTETS_PACKETS_GOOD_BAD	TX128_255OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0730	ETH_TX_256TO511OCTETS_PACKETS_GOOD_BAD	TX256_511OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0734	ETH_TX_512TO1023OCTETS_PACKETS_GOOD_BAD	TX512_1023OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0738	ETH_TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD	TX1024_MAXOCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x073C	ETH_TX_UNICAST_PACKETS_GOOD_BAD	TXUCASTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0740	ETH_TX_MULTICAST_PACKETS_GOOD_BAD	TXMCASTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0744	ETH_TX_BROADCAST_PACKETS_GOOD_BAD	TXBCASTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0748	ETH_TX_UNDERFLOW_ERROR_PACKETS	TXUNDRFLW[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x074C	ETH_TX_SINGLE_COLLISION_GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0750	ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754	ETH_TX_DEFERRED_PACKETS	TXDEFRD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0758	ETH_TX_LATE_COLLISION_PACKETS	TXLATECOL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x075C	ETH_TX_EXCESSIVE_COLLISION_PACKETS	TXEXSCOL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0760	ETH_TX_CARRIER_ERROR_PACKETS	TXCARR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0764	ETH_TX_OCTET_COUNT_GOOD	TXOCTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0768	ETH_TX_PACKET_COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C	ETH_TX_EXCESSIVE_DEFERRAL_ERROR	TXEXSDEF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0770	ETH_TX_PAUSE_PACKETS	TXPAUSE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0774	ETH_TX_VLAN_PACKETS_GOOD	TXVLANG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0778	ETH_TX_OSIZE_PACKETS_GOOD	TXOSIZG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x077C	Reserved																																
0x0780	ETH_RX_PACKETS_COUNT_GOOD_BAD	RXPKTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0784	ETH_RX_OCTET_COUNT_GOOD_BAD	RXOCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0788	ETH_RX_OCTET_COUNT_GOOD	RXOCTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078C	ETH_RX_BROADCAST_PACKETS_GOOD	RXBCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0790	ETH_RX_MULTICAST_PACKETS_GOOD	RXMCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0794	ETH_RX_CRC_ERROR_PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	ETH_RX_ALIGNMENT_ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C	ETH_RX_RUNT_ERROR_PACKETS	RXRUNTERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07A0	ETH_RX_JABBER_ERROR_PACKETS	RXJABERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07A4	ETH_RX_UNDERSIZE_PACKETS_GOOD	RXUNDERSZG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07A8	ETH_RX_OVERSIZE_PACKETS_GOOD	RXOVERSZG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07AC	ETH_RX_64OCTETS_PACKETS_GOOD_BAD	RX64OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07B0	ETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD	TX65_127OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07B4	ETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD	RX128_255OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07B8	ETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD	RX256_511OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07BC	ETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD	RX512_1023OCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C0	ETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD	RX1024_MAXOCTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C4	ETH_RX_UNICAST_PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8	ETH_RX_LENGTH_ERROR_PACKETS	RXLENERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07CC	ETH_RX_OUT_OF_RANGE_PACKETS	RXOUTOFRNG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07D0	ETH_RX_PAUSE_PACKETS	RXPAUSEPKT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07D4	ETH_RX_FIFO_OVERFLOW_PACKETS	RXFIFOOVFL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07D8	ETH_RX_VLAN_PACKETS_GOOD_BAD	RXVLANPKTGB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07DC	ETH_RX_WATCHDOG_ERROR_PACKETS	RXWDGERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07E0	ETH_RX_RECEIVE_ERROR	RXRCVERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07E4	ETH_RX_CONTROL_PACKETS_GOOD	RXCTRLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07E8	Reserved																																
0x07EC	ETH_TX_LPI_USEC_CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07F0	ETH_TX_LPI_TRAN_CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	ETH_RX_LPI_USEC_CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	ETH_RX_LPI_TRAN_CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07FC-0x89C	Reserved																																
0x08A0	ETH_MMC_FPE_TX_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIS	FCIS
	Reset value																															0	0
0x08A4	ETH_MMC_FPE_TX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIM	FCIM
	Reset value																															0	0
0x08A8	ETH_MMC_FPE_TX_FCR	TXFFC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08AC	ETH_MMC_TX_HRCR	TXHRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08B0-0x08BC	Reserved																																
0x08C0	ETH_MMC_FPE_RX_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIS	PAOCIS	PSECIS	PAECIS
	Reset value																													0	0	0	0
0x08C4	ETH_MMC_FPE_RX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIM	PAOCIM	PSECIM	PAECIM
	Reset value																													0	0	0	0
0x08C8	ETH_RX_PACKET_ASM_ERR	PAEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08CC	ETH_RX_PACKET_SMD_ERR	PSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08D0	ETH_RX_PACKET_ASM_OKR	PAOC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08D4	ETH_RX_FPE_FRAG_CR	FFC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08D8-0x08FC	Reserved																																
0x0900	ETH_ MACL3L4C0R	Res.	Res.	Res.	DMCHEN0	Res.	Res.	Res.	DMCHN0	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0	L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
	Reset value				0				0			0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_ MACL3L4C1R	Res.	Res.	Res.	DMCHEN1	Res.	Res.	Res.	DMCHN1	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1	L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
	Reset value				0				0			0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0950 - 0x0A6C	Reserved																																

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A70	ETH_MAC_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSEL[3:0]				AOFF[7:0]								Res.	Res.	AUTO	Res.	Res.	Res.	COM.	OB
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0			0				0	0
0x0A74	ETH_MAC_ TMRQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PFEX	Res.	TMRW[2:0]			TYP[15:0]																	
	Reset value										0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A78 - 0x0AFC	Reserved																																
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	ESTI	Res.	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0				0		0	0	0	0	0	1	0	0	0	0	0		0	0		0	0	0	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR1	TSTARGT1	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0										0	0	0	0	0	0
0x0B24	ETH_MACRXDTI	RXNS[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B28	ETH_MACTXDTI	TXNS[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x0B2C	Reserved																																
0x0B30	ETH_ MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B34	ETH_ MACTXTSSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																
0x0B48	ETH_MACATSNR	Res.	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_ MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B64	Reserved																																
0x0B68	ETH_ MACTSILR	Res.	Res.	Res.	Res.	ITLNS[11:0]												ITLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B6C	ETH_ MACTSELR	Res.	Res.	Res.	Res.	ETLNS[11:0]												ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value				0																					0	0	0	0	0	0	0	0
0x0B70	ETH_MACPPSCR (alternate)	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN1	TRGTMODSEL1[1:0]		Res.	PPSCMD1[3:0]				MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value				0													0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTS0R	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTN0R	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSI0R	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8C	ETH_MACPPSW0R	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90	ETH_ MACPPSTTS1R	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B94	ETH_ MACPPSTTN1R	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B98	ETH_MACPPSI1R	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B9C	ETH_MACPPSW1R	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BA0 - 0x0BBC	Reserved																																

Table 888. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BC0	ETH_MACPOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DN[7:0]								PRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BCC	ETH_MACSPI2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.4: Memory organization* for the register boundary addresses.

78 Hardware debug port (HDP)

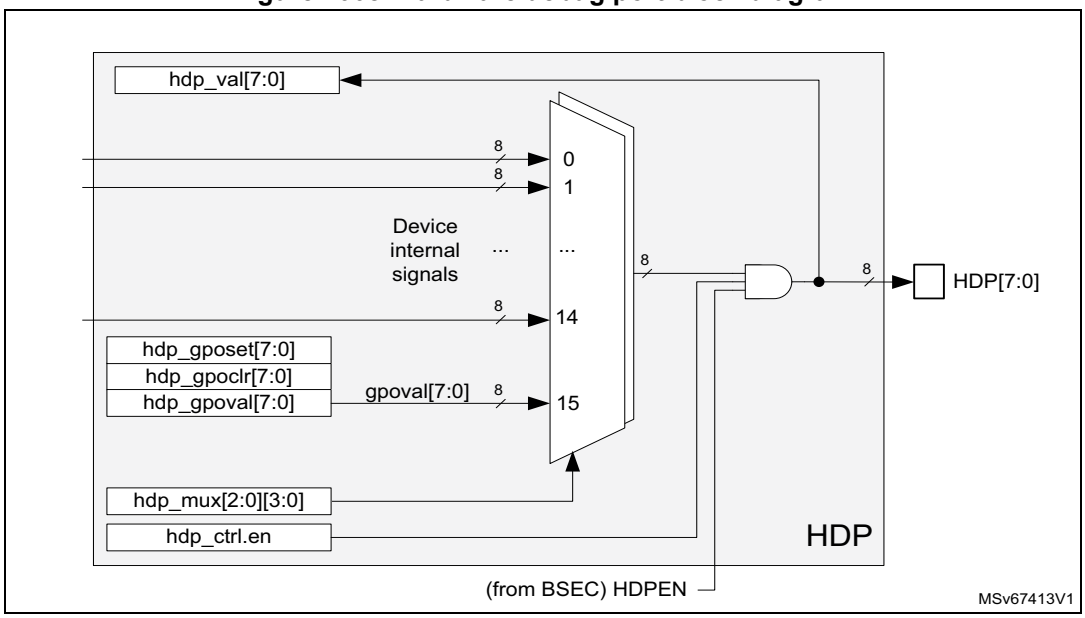
The hardware debug port allows the observation of internal signals. By using multiplexers, up to 16 signals for each of 8-bit output can be observed.

78.1 HDP main features

- 8 output signals
- One of 15 internal signals with individual control
- 8 software programmable signals for pinout agnostic code debugging
- Output disabling by security signal

78.2 Functional description

Figure 1008. Hardware debug port block diagram



The control of the main multiplexer that allows the selection of the 8-signal debug set to be output is done through the HDP_CTRL register.

The HDP value at the output of the main multiplexer can be read by software through the HDP_VAL register.

During debug, it is often useful to output some signals that are controlled by the software. They can be used to observe precisely a code sequence, or to indicate the current executed task. The debug set #15 is reserved for this purpose: it provides on HDP outputs, HDP_GPOVAL value that is associated with HDP_GPOSET and HDP_GPOCLR, allowing the signals to be toggled without a read-modify-write sequence.

Some debug sets expose internal data that require to be protected: the HDPEN signal, when low, forces all the HDP block outputs low.

Note: The internal signals may toggle at a higher rate than the pins can support. Their observability may then be compromised. The user must be aware of this limitation when using the HDP port.

Table 889. HDP signal muxing

HDP_MUX	HDP7	HDP6	HDP5	HDP4	HDP3	HDP2	HDP1	HDP0
	MUX7[3:0]	MUX6[3:0]	MUX5[3:0]	MUX4[3:0]	MUX3[3:0]	MUX2[3:0]	MUX1[3:0]	MUX0[3:0]
0	CPU1 STANDBY WFI0	-	CPU1 STANDBY WFIL2	PWR SEL_VTH_ VDDCORE	PWR SEL_VTH_ VDDCPU	PWR pwrwake_ cpu1	PWR pwrwake_ cpu2	PWR pwrwake_ sys
1	CPU1 STANDBY WFE0	-	-	CPU2 SLEEPING	CPU2 TXEV	CPU2 RXEV	CPU2 HALTED	CPU2 SLEEP DEEP
2	CPU1 HALTED0	-	-	CPU1 EVENTO	CPU1 nPMUIRQ0	-	CPU2 State[1]	BSEC out test_sdr _unlock OR DISABLE_S CAN
3	-	CPU1 nAXIER RIRQ	CPU1 nIRQOUT0	-	CPU1 nFIQOUT0	-	BSECout dbgenm	BSEC out nidenm
4	BSEC out spidenm	BSEC out spnidenm	BSEC out spidena	BSEC out spnidena	BSEC out ddbgen	BSEC out shdbgen	BSEC out dbgena	BSEC out nidena
5	EXTI2 cpu1 _wakeup	EXTI2 cpu2 _wakeup	-	EXTI2 sys_wakeup	EXTI1 cpu1 _wakeup	EXTI1 cpu2 _wakeup	EXTI1 sys_ wakeup	CPU2 State[0]
6	-	-	-	-	-	RCC pwrds _cpu1	RCC pwrds _cpu2	RCC pwrds _sys
7	-	-	-	-	-	-	-	-
8	DDRSS dfi_ctrlupd _req	DDRSS dfi_init _complete	DDRCTRL dfi_init_start	DDRSS dphycg_en	DDRSS mcdcg_en	DDRSS cactive _ddrc	DDRSS csysack _ddrc	DDRSS csysreq _ddrc
9	DDRSS obsp3	DDRSS obsp2	DDRSS obsp1	DDRSS obsp0	DDRSS dfi_freq[0]	DDRSS dfi_lp_req	DDRSS dfi_phymstr _req	DDRSS dfi_phyupd _req
10	D2 State[1]	D2 State[0]	pll6_lock	pll5_lock	pll4_lock	-	pll2_lock	pll1_lock
11	pll8_lock	pll7_lock	-	HPDMA3 clk_bus_req	HPDMA2 clk_bus_req	HPDMA1 clk_bus_req	D2 gbl_per _dma_req	D2 gbl_per _clk_bus _req
12	USB_HDP[7]	USB_HDP[6]	USB_HDP[5]	USB_HDP[4]	USB_HDP[3]	USB_HDP[2]	USB_HDP[1]	USB_HDP[0]
13	-	octospi1_dll _lock	USB_HDP[1 3]	USB_HDP[1 2]	USB_HDP[1 1]	USB_HDP[10]	USB_HDP[9]	USB_HDP[8]

Table 889. HDP signal muxing (continued)

HDP_MUX	HDP7	HDP6	HDP5	HDP4	HDP3	HDP2	HDP1	HDP0
	MUX7[3:0]	MUX6[3:0]	MUX5[3:0]	MUX4[3:0]	MUX3[3:0]	MUX2[3:0]	MUX1[3:0]	MUX0[3:0]
14	CPU1 State[1]	CPU1 State[0]	D1 State[2]	D1 State[1]	D1 State[0]	sdmmc3_dll_lock	sdmmc2_dll_lock	sdmmc1_dll_lock
15	gpoval[7]	gpoval[6]	gpoval[5]	gpoval[4]	gpoval[3]	gpoval[2]	gpoval[1]	gpoval[0]

78.3 HDP registers

78.3.1 HDP control register (HDP_CTRL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: HDP enable (valid if enabled in BSEC)

78.3.2 HDP multiplexer control register (HDP_MUX)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MUXy[3:0]**: Selection of the HDPy output among the 16 available signals (y = 7 to 0)

78.3.3 HDP read back value register (HDP_VAL)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPVAL[7:0]**: Value of the HDP signals

78.3.4 HDP general purpose output set register (HDP_GPOSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOSET[7:0]**:

When a bit is written to 1, the corresponding HDP GPO is set.
Writing a bit to 0 has no effect.

78.3.5 HDP general purpose output clear register (HDP_GPOCLR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOCLR[7:0]**:
When a bit is written to 1, the corresponding HDP GPO is cleared.
Writing a bit to 0 has no effect.

78.3.6 HDP general purpose output value register (HDP_GPOVAL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOVAL[7:0]**:
When written, this bitfield defines the value of the HDP GPO.
When read, this bitfield provides the current GPO value.

78.3.7 HDP version register (HDP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** HDP major revision

Bits 3:0 **MINREV[3:0]:** HDP minor revision

78.3.8 HDP identification register (HDP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0003 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: HDP identifier

78.3.9 HDP size identification register (HDP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identifier

78.3.10 HDP register map

Table 890. HDP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HDP_CTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x004	**HDP_MUX**	MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]				MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008-0x00C	Reserved	Res.erved																															
0x010	**HDP_VAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	**HDP_GPOSET**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 890. HDP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	**HDP_GPOCLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	**HDP_GPOVAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x020-0x3F0	Reserved	Res.erved																															
0x3F4	**HDP_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	**HDP_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	**HDP_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

79 Software error robustness collector (SERC)

79.1 Introduction

The SERF (software-error robustness framework) detects accesses to target peripherals that are either shutdown (computing clock off), or under reset. Upon detection of the access, the software-error robustness detector (SERD) answers the bus request instead of the shutdown target. It reports a bus illegal access (bus_ilac) to the software-error robustness collector (SERC). In this context, a bus illegal access is an access to peripherals either under reset, or with clock not enabled.

The SERC fires an interrupt, and is equipped with registers to identify the target generating the fault, and enabling/masking corresponding interrupt.

The primary use of the SERF is to prevent CPU hangs on accessing AXI shutdown targets.

The SERF works in collaboration with the RCC, which provides clocks and resets.

79.2 SERF main features

- Cover all targets, provided with either AXI or AHB interfaces
- Reply (with error) to requests directed to shut down AXI targets, preventing CPU hangs
- Notify through interrupts to the CPU that access to a shutdown/under-reset target has been attempted
- Identify the unavailable target registered

The SERF includes:

- a set of detectors (SERD)
- a collector (SERC), provided with an APB interface, which gathers all detected error sources, fires an interrupt (if needed), and logs the detector ID in a local register

The detectors have a zero-cycle pass through latency if the target is enabled.

79.3 SERF functional description

79.3.1 SERF block diagram

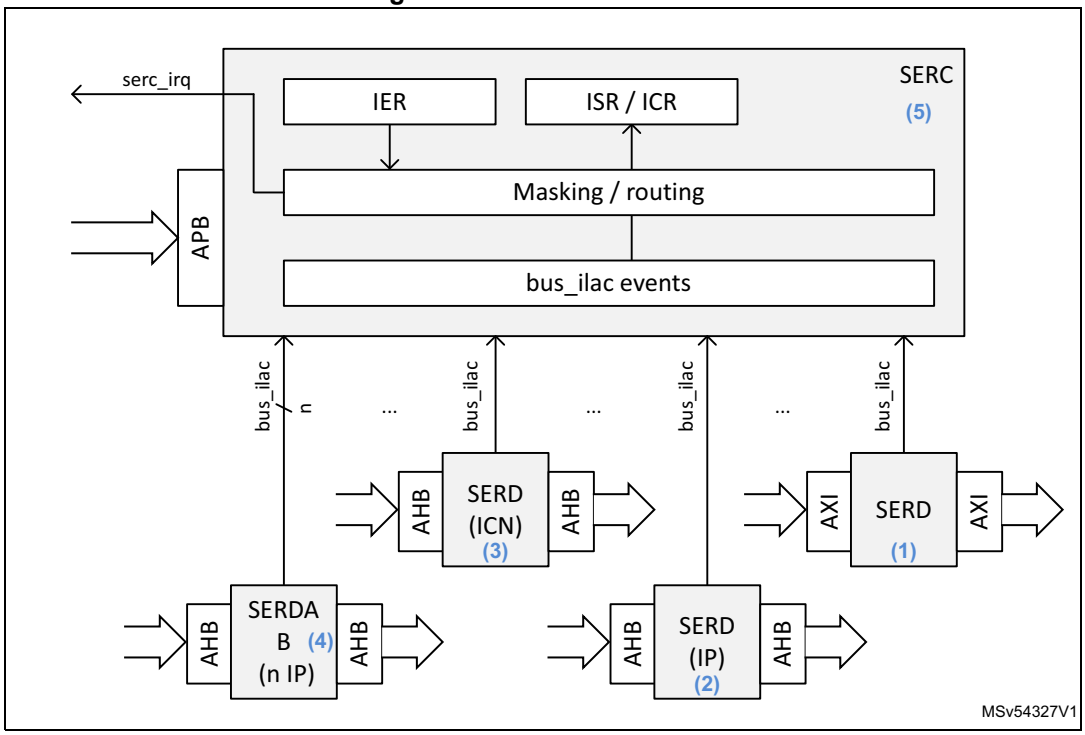
The SERF is made up of four building blocks, namely AXI SERD, AHB SERD, AHB SERDAB (SERD address based), and SERC.

AXI SERD, AHB SERD, and AHB SERDAB can be instantiated as many times as required, while usually there is a single SERC instance.

Figure 1009 shows a SERF example interconnect featuring:

- One SERD instance (1) with AXI interfaces, located in front of an AXI target peripheral
- One SERD instance (2) with AHB interfaces, located in front on an AHB target peripheral
- One SERD instance (3) with AHB interfaces, located in front of an AHB grape interconnect
- One SERDAB instance (4) with AHB interfaces, located in front of an AHB2APB grape interconnect
- One SERC instance (5) collecting the bus_ilac reported by the detectors, and generating interrupts. The SERC is provided with an APB interface to access its control registers.

Figure 1009. SERF overview



When a SERD gets a bus access to a target shutdown or under reset, it generates an error response on the data bus and a bus_ilac to the SERC. Each bus_ilac is recorded in SERC_ISRx registers, and a serc_irq is fired.

The interrupt handler can do the following:

- Read SERC_ISRx to understand which peripheral was accessed while under reset or shutdown.
- Clear this status (and as well the interrupt) through SERC_ICRx.
- Mask an interrupt generated by an illegal access from a specific peripheral through SERC_IECx.

79.3.2 SERF key internal signals

Table 891. Key internal signals

Module	Interfaces	Functionality
AXI SERD	One AXI slave and one AXI master	Receives the clk_en_ip and rstn_ip of the peripheral. It is connected to from the RCC. Reports a bus_ilac to the SERC.
AHB SERD	One AHB slave and one AHB master	Receives the clk_en_ip and rstn_ip of the peripheral. It is connected to from the RCC. If the SERD is connected to an AHB matrix, it receives the clk_en_ip and the rsnt_ip of all the IPs connected to the AHB matrix. Reports a bus_ilac to the SERC.
AHB SERDAB		Receives the clk_en_ip and rstn_ip of all the peripherals connected to the AHB2APB downstream from the RCC. Reports a bus_ilac to the SERC per address region it protects.
SERC	One slave APB	Receives all bus_ilac signals from the embedded SERD /SERDAB. Generates a SERC_irq when one or several bus_ilac are detected.

The SERC resource allocation is aligned to the one defined for the RIF (product dependent).

79.3.3 SERF resets and clocks

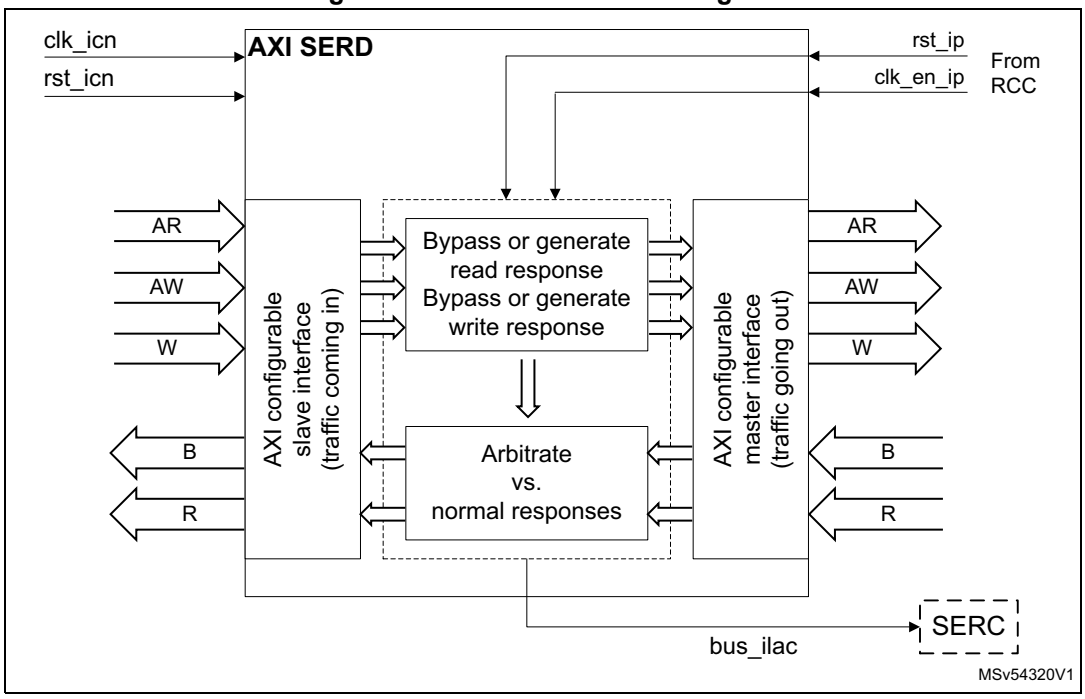
AXI SERD, AHB SERD, and AHB SERDAB receive the same clock and reset signals of the interconnect they are inserted in. SERC has a single clock and a single reset. All clocks and resets are provided by the RCC.

79.3.4 Functionality and programming model

AXI SERD

As shown in *Figure 1010*, AXI SERD is made up of one AXI slave interface, one AXI master interface, and an internal logic to detect access to non-available target peripheral, generating appropriate response, and reporting a bus_ilac to the SERC.

Figure 1010. AXI SERD block diagram

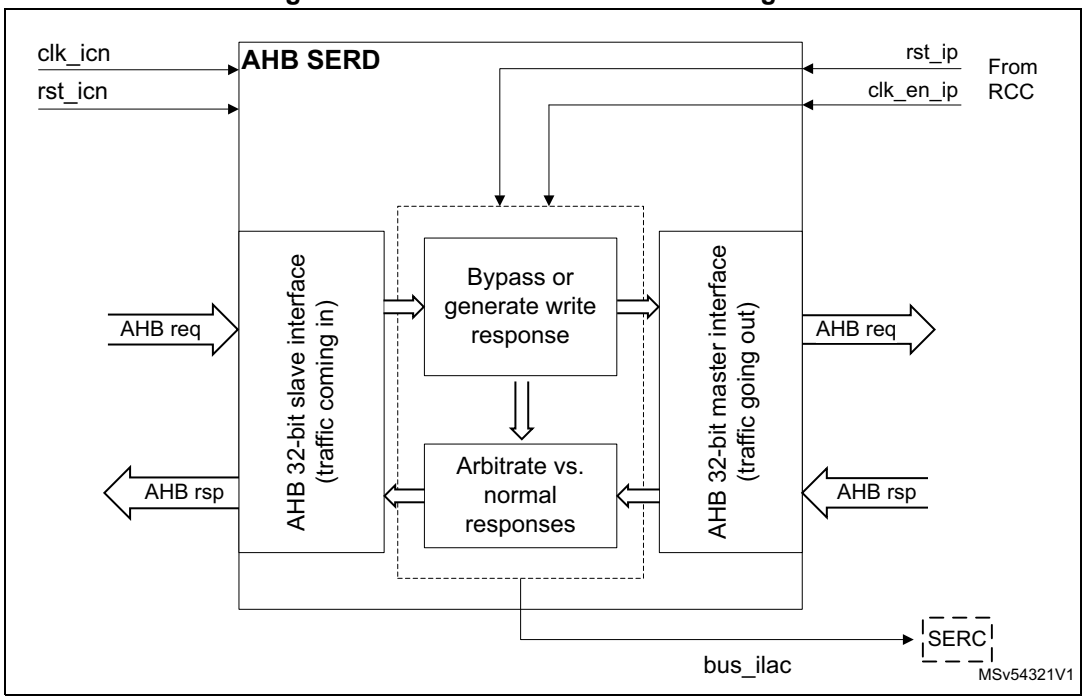


Note: The AXI SERD does not allow recover when a peripheral is wrongly shut down (when there are still outstanding responses expected from the peripheral). It only prevents that a peripheral under shutdown or reset is accessed.

AHB SERD

Figure 1011 shows a block diagram of the AHB SERD (for IP). The AHB SERD is made up of one AHB slave interface, one AHB master interface, and an internal logic for detecting access to non available target IP, generating an appropriate response, and reporting a bus_ilac to the SERC.

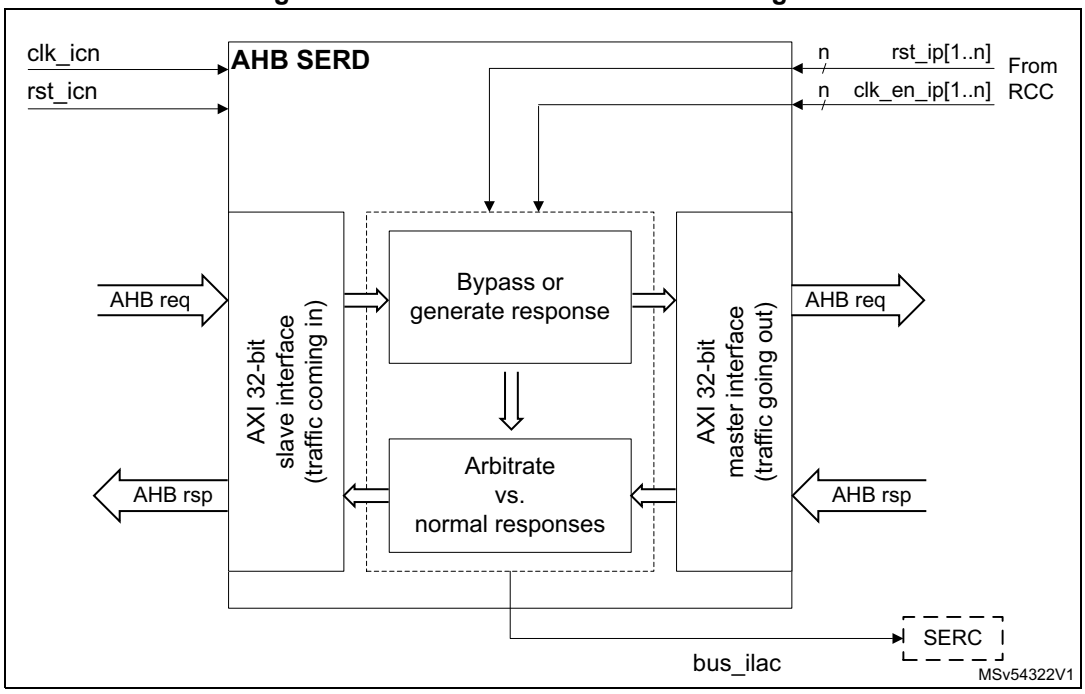
Figure 1011. AHB SERD for IP block diagram



As shown in *Figure 1012*, the AHB SERD is made up of one AHB slave interface, one AHB master interface, and an internal logic for detecting access to non-available target peripherals, generating an appropriate response, and reporting a bus_ilac to the SERC.

The AHB SERD (for ICN) differs from the AHB SERD (for peripheral) because it is located in front of an AHB grape. If all targets connected to this grape are switched off, the grape is switched off as well. The AHB SERD for ICN must handle any access directed to it. For this purpose, it receives all ip_clk_en and ip_rst signals of the targets connected to the AHB grape it protects.

Figure 1012. AHB SERD for ICN block diagram

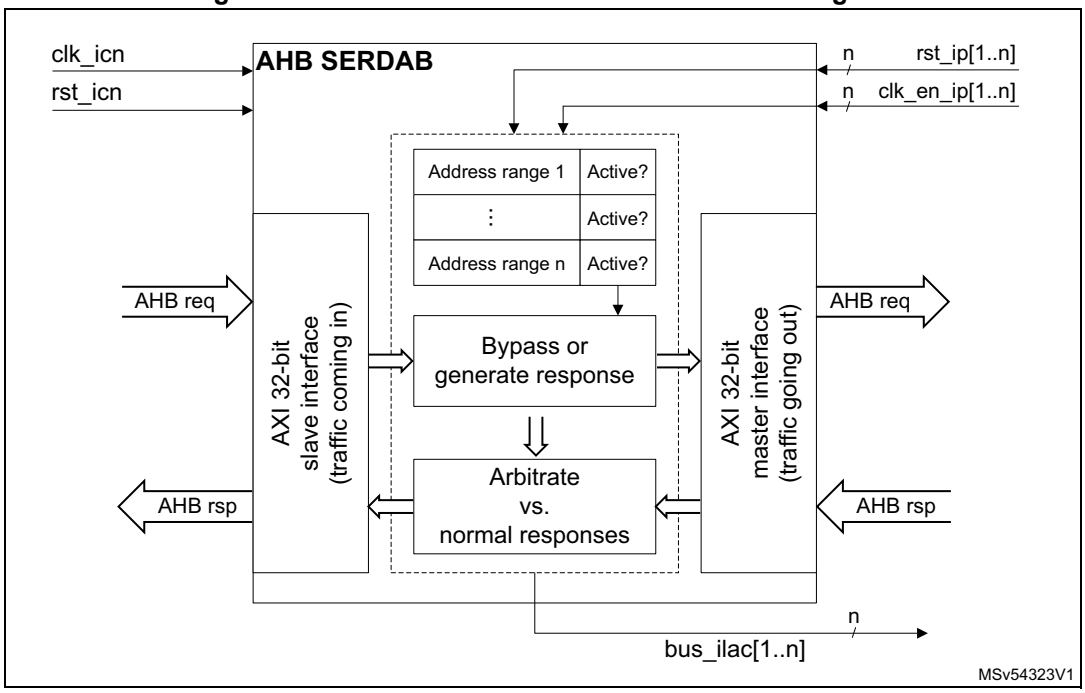


AHB SERDAB

Figure 1013 shows the block diagram of the AHB SERDAB for shielding an APB2AHB bridge. The AHB SERDAB is made up of one AHB slave interface, one AHB master interface, and an internal logic for detecting access to non-available target peripherals, generating appropriate response, and reporting a bus_ilac to the SERC. The AHB SERDAB differs from the AHB SERD (for peripheral) because it is located in front of an AHB2APB bridge. Each APB target connected to the AHB2APB bridge can drive its pready and pslverr signals or not. If the target does not drive pready and pslverr, these signals are tied, respectively, to 1 and 0. It is then not possible to discriminate if the response at the AHB master interface comes really from the APB target, or is a dummy response, from a target that may be in power down. To overcome this limitation, the AHB SERDAB can do the following:

- It receives from the RCC all the clk_en_ip and rst_ip of the targets connected to the AHB2APB bridge.
- It is provided with an internal address decoder, and each region is active or not depending on the corresponding clk_en_ip and rst_ip.
- When a transaction address is detected in an address range inactive, the corresponding bus_ilac signal is reported to the SERC.

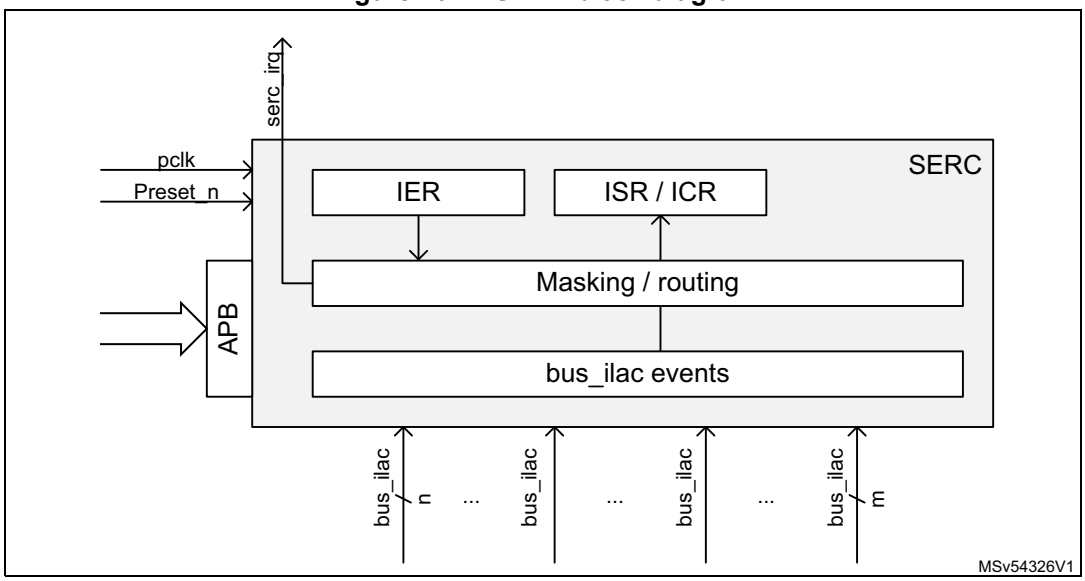
Figure 1013. AHB SERDAB for AHB2APB block diagram



SERC

The SERC gathers in a central place all the bus_ilac signals and registers to enable, mask, and clear the resulting serc_irq. The SERC maps each bus_ilac to one of its registers, accessible through an APB interface.

Figure 1014. SERC block diagram



The device embeds a RIF infrastructure, hence a RISUP is present in front of the APB interface. *Figure 1015* shows the SERF when connected to the RISUP block.

Figure 1015. SERF with an RISUP

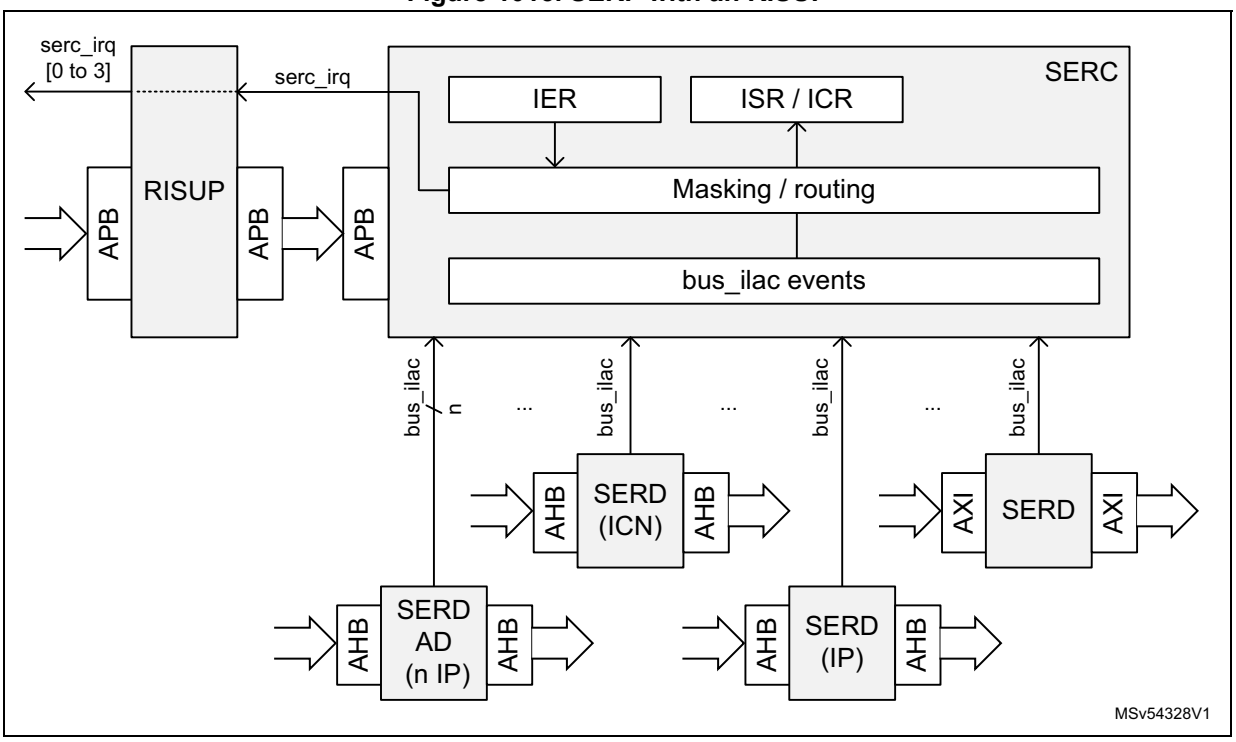


Table 892. SERF resources mapping

Peripherals 31 to 0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USART1	SPDIFRX	-	-	SPI6	SPI5	SPI4	SPI3	SPI2	SPI1	LPTIM5	LPTIM4	LPTIM3	LPTIM2	LPTIM1	TIM20	TIM17	TIM16	TIM15	TIM14	TIM13	TIM12	TIM11	TIM10	TIM8	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2	TIM1
Peripherals 63 to 32																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
USBH	To AHB6	ETH2	ETH1	ADC2	ADC1	HDP	FDCAN	-	MDF1	-	SAI4	SAI3	SAI2	SAI1	-	-	-	-	-	I2C3	I2C2	I2C1	LPUART1	-	-	UART7	USART6	UART5	UART4	USART3	USART2
Peripherals 95 to 64																															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
SAES	PKA	RNG2	RNG1	-	-	-	DCMI PSSI	DCMIPP	CSI	-	-	-	-	-	LTDC_CMN	-	SDMMC3	SDMMC2	SDMMC1	OCTOSPI2	OCTOSPI1	STGEN	-	-	-	-	-	DDRPERFM	OTG_HS	-	-

Table 892. SERF resources mapping (continued)

Peripherals 127 to 96																																
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
IAC	-	OTFDEC1	-	-	-	-	LTDC_L3	LTDC_L1L2	ICACHE DCACHE	-	I3C3	I3C2	I3C1	-	-	-	SERC	CRC	-	DTS	VREFBUF	-	WWDG1	IWDG4	IWDG3	IWDG2	IWDG1	CRYP2	CRYP1	HASH2	HASH1
Peripherals 159 to 128																																
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
SYSCFG	RISC	CA35SS	RCC	PWR	EXTI2	EXTI1	TAMP	RTC	-	HPDMA3	HPDMA2	HPDMA1	-	-	IPCC1	-	-	-	MCE	-	-	RISAF4 or DDRCTRL port1 or DDRCTRL	-	RISAF2	RISAF1 or BKPSRAM	-	RISAB5 or RETRAM	-	RISAB3 or SRAM1	RISAB2	RISAB1 or SYSRAM
Peripherals 191 to 160																																
191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
To AHB3	To AHB5	STM	APBDEBUG	-	To SDMMCconf	To USBconf	-	-	To AHB2	-	-	-	DBGMCU	BSEC	FMC	DDRPHYC	DDRCTRL or DDRDBG	-	-	GPIOZ	-	-	GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA

79.4 SERF low-power modes

The SERF does not support low power modes.

79.5 SERC interrupts

The SERC fires an interrupt when a detector reports a bus_ilac to it. The interrupt is cleared by writing the SER_ICR register matching the SERC_ISR that generated the interrupt. The SERC_ICR auto resets itself when the corresponding SERC_ISR bit is reset.

The interrupt remains high as long as there is one bit set to 1 in the SERC_ISR register. This ensures that none of the reported bus_ilac cam be missed by the interrupt handler.

Interrupts can be masked individually for each bus_ilac source through register SERC_IER.

79.6 SERC registers

The AXI/AHB SERD/SERDAB are not provided with registers.

The SERC has a set of registers:

- to log the target peripheral that has generated an interrupt
- to enable or mask the generation of the interrupt for each single target peripheral
- to clear the interrupt and reset the status of the logged target peripheral

SERC is disabled at boot time, and must be enabled.

On top of these functional registers, the SERC is also provided with configuration registers:

- hardware configuration registers, embedding configured values of design generics
- version, size, and identification registers

All SERC registers are accessed by words (32-bit). Byte and 16-bit write accesses are ignored and have no effect, read accesses always return 0. In all cases, byte and 16-bit accesses do not generate a bus error.

Read accesses to unmapped addresses or write-only registers always report errors. Write accesses to unmapped accesses or read-only registers are ignored and have no effects.

- SERC_IERx registers are used to enable an interrupt when an access to non-available (either shutdown or under reset) target occurs. These registers are accessible in read or write.
- SERC_ISRx registers are used to read the status of all accesses to non-available (either shutdown or under reset) events. These registers are read only.

  The collector asserts the bit corresponding to a bus_ilac input and fires an interrupt. Upon reception of the interrupt, the interrupt handler can check this register to retrieve which peripheral generated the interrupt.
- SERC_ICRx registers are used to clear a pending illegal access interrupt. These registers are write only.

  When the interrupt handler has finished dealing with the power-down or reset illegal access interrupt, it writes in this register to clear the interrupt and reset the status in the corresponding peripheral SERC_ISRx register. This register is then auto-reset.

79.6.1 SERC interrupt enable register 0 (SERC_IER0)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[31:0]**: Power-down or reset interrupt enable for peripheral y (y = 0 to 31)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.2 SERC interrupt enable register 1 (SERC_IER1)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[63:32]**: Power-down or reset interrupt enable for peripheral y (y = 32 to 63)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.3 SERC interrupt enable register 2 (SERC_IER2)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[95:64]**: Power-down or reset interrupt enable for peripheral y (y = 64 to 95)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.4 SERC interrupt enable register 3 (SERC_IER3)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[127:96]**: Power-down or reset interrupt enable for peripheral y (y = 96 to 127)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.5 SERC interrupt enable register 4 (SERC_IER4)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[159:144]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[143:128]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[159:128]**: Power-down or reset interrupt enable for peripheral y (y = 128 to 159)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.6 SERC interrupt enable register 5 (SERC_IER5)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRIE[191:176]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRIE[175:160]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PDRIE[191:160]**: Power-down or reset interrupt enable for peripheral y (y = 160 to 191)
Sets the bit to unmask access to power-down or reset events from the peripheral y.
0: power-down or reset access event from peripheral y does not generate an interrupt (masked)
1: power-down or reset access event from peripheral y generates an interrupt (unmasked)

79.6.7 SERC interrupt status register 0 (SERC_ISR0)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[31:0]**: Power-down or reset interrupt status for peripheral y (y = 0 to 31)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.8 SERC interrupt status register 1 (SERC_ISR1)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[63:32]**: Power-down or reset interrupt status for peripheral y (y = 32 to 63)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.9 SERC interrupt status register 2 (SERC_ISR2)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[95:64]**: Power-down or reset interrupt status for peripheral y (y = 64 to 95)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.10 SERC interrupt status register 3 (SERC_ISR3)

Address offset: 0x004C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[127:112]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[111:96]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[127:96]**: Power-down or reset interrupt status for peripheral y (y = 96 to 127)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.11 SERC interrupt status register 4 (SERC_ISR4)

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[159:144]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[143:128]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[159:128]**: Power-down or reset interrupt status for peripheral y (y = 128 to 159)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.12 SERC interrupt status register 5 (SERC_ISR5)

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRS[191:176]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRS[175:160]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PDRS[191:160]**: Power-down or reset clear flag for peripheral y (y = 160 to 191)
The collector sets this bit when a bus_ilac is collected for peripheral y.
0: no power-down or reset event recorded for this peripheral
1: a power-down or reset access event has been reported from this peripheral

79.6.13 SERC interrupt clear register 0 (SERC_ICR0)

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[31:0]**: Power-down or reset clear flag for peripheral y (y = 0 to 31)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR0 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR0 register

79.6.14 SERC interrupt clear register 1 (SERC_ICR1)

Address offset: 0x0084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[63:32]**: Power-down or reset clear flag for peripheral y (y = 32 to 63)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR1 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR1 register

79.6.15 SERC interrupt clear register 2 (SERC_ICR2)

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[95:64]**: Power-down or reset clear flag for peripheral y (y = 64 to 95)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR2 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR2 register

79.6.16 SERC interrupt clear register 3 (SERC_ICR3)

Address offset: 0x008C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[127:96]**: Power-down or reset clear flag for peripheral y (y = 96 to 127)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR3 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR3 register

79.6.17 SERC interrupt clear register 4 (SERC_ICR4)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[159:128]**: Power-down or reset clear flag for peripheral y (y = 128 to 159)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR4 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR4 register

79.6.18 SERC interrupt clear register 5 (SERC_ICR5)

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDRC[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDRC[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDRC[191:160]**: Power-down or reset clear flag for peripheral y (y = 160 to 191)
When set, this write-only bit clears the status flag of the illegal access event y set in SERC_ISR5 register.
0: PDRCy flag status is not affected
1: PDRCy flag status is cleared in SERC_ISR5 register

79.6.19 SERC enable (SERC_ENABLE)

Address offset: 0x0100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SERFEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SERFEN**: Enable the SERF framework.

When the SERF is enabled, an access to a target under reset or clocked off is detected; a functional error response is given back on the IP bus, an illegal access is reported to the SERC, and an interrupt is fired according to the SERC registers configuration. When the SERF is disabled, an access to a target under reset or clocked off is not detected.

79.6.20 SERC hardware configuration register (SERC_HWCFGR)

Address offset: 0x03F0

Reset value: 0x000X 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **CFG2[2:0]**: NUM_RESYNC_FLOPS, number of resynchronization flops when receiving a bus_ilac information in the SERC clock domain.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **CFG1[7:0]**: NUM_ILAC, number of illegal accesses input pins configured for this instance of the collector.

79.6.21 SERC version register (SERC_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

79.6.22 SERC identification register (SERC_IPIDR)

Address offset: 0x03F8

Reset value: 0x0010 01F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code

79.6.23 SERC size identification register (SERC_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code (1 Kbyte)

79.6.24 SERC register map

Table 893. SERC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SERC_IER0	PDRIE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	SERC_IER1	PDRIE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SERC_IER2	PDRIE[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SERC_IER3	PDRIE[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SERC_IER4	PDRIE[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SERC_IER5	PDRIE[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	SERC_ISR0	PDRS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SERC_ISR1	PDRS[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SERC_ISR2	PDRS[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SERC_ISR3	PDRS[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SERC_ISR4	PDRS[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SERC_ISR5	PDRS[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	SERC_ICR0	PDRS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	SERC_ICR1	PDRS[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	SERC_ICR2	PDRS[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	SERC_ICR3	PDRS[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	SERC_ICR4	PDRS[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	SERC_ICR5	PDRS[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	SERC_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SERFEN
	Reset value																																0
0x104-0x3EC	Reserved	Reserved																															
0x3F0	SERC_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG1[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x3F4	SERC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	SERC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0x3FC	SERC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1

Refer to *Section 2.4* for the register boundary addresses.

80 Debug support (DBG)

80.1 DBG introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/serial-wire debug access port, using industry standard debugging tools.

A trace port allows data to be captured for logging and analysis.

The debug features are based on the following Arm CoreSight components:

- SWJ-DP: JTAG/serial-wire debug port
- AXI-AP: AXI access port
- AHB-AP: AHB access port
- APB-AP: APB access port
- ITM: instrumentation trace macrocell
- DWT: data watchpoint and trace
- ETM: embedded trace macrocell
- ETF: embedded trace FIFO
- ETR: embedded trace router
- TPIU: trace port interface unit
- SWO: serial wire output
- CTI: cross trigger interface
- CTM: cross trigger matrix
- Timestamp generator
- STM: system trace macrocell

These components are described in *Section 80.9*. More information can be found in the Arm documents referenced in *Section 80.15*.

80.2 DBG use cases

The trace and debug system are designed to support a variety of typical use cases:

- Low cost software trace (Cortex-M33 only)

  A limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling.

- External debugging of each core independently

  All processor cores can be simultaneously and independently debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint settings, code stepping, and memory access.
- Self-hosted debugging of each core independently

  All processor cores can be simultaneously and independently debugged by software running on one of the cores. This allows breakpoint and watchpoint settings, code stepping, and memory access.
- Synchronous debugging of cores

  When one core stops due to a breakpoint or a debugger stop command, other cores can be stopped as well. Similarly, the cores can be restarted at the same time. This allows debugging of loosely coupled applications which require the processors to remain synchronized.
- Tracing code execution from one or more cores via the trace port

  The trace information from all cores is combined into a single trace stream and output to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
- Code and hardware instrumentation via the trace port

  The software data from the processors, memory content via DMA, and hardware event trace information can be generated using the STM. The information is combined with the processor trace stream and output to a trace port analyzer in real time.
- Capturing trace continuously into the ETF programmed as a circular buffer (ETB)

  The combined trace information can be stored on-chip in a private circular buffer. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, or an internal event.
- Capturing trace continuously into a system memory

  Thanks to the ETR, the combined trace information can be sent to a system memory, either on-chip SRAM or off-chip DDR. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, an internal event.
- Reading the stored trace with the debugger

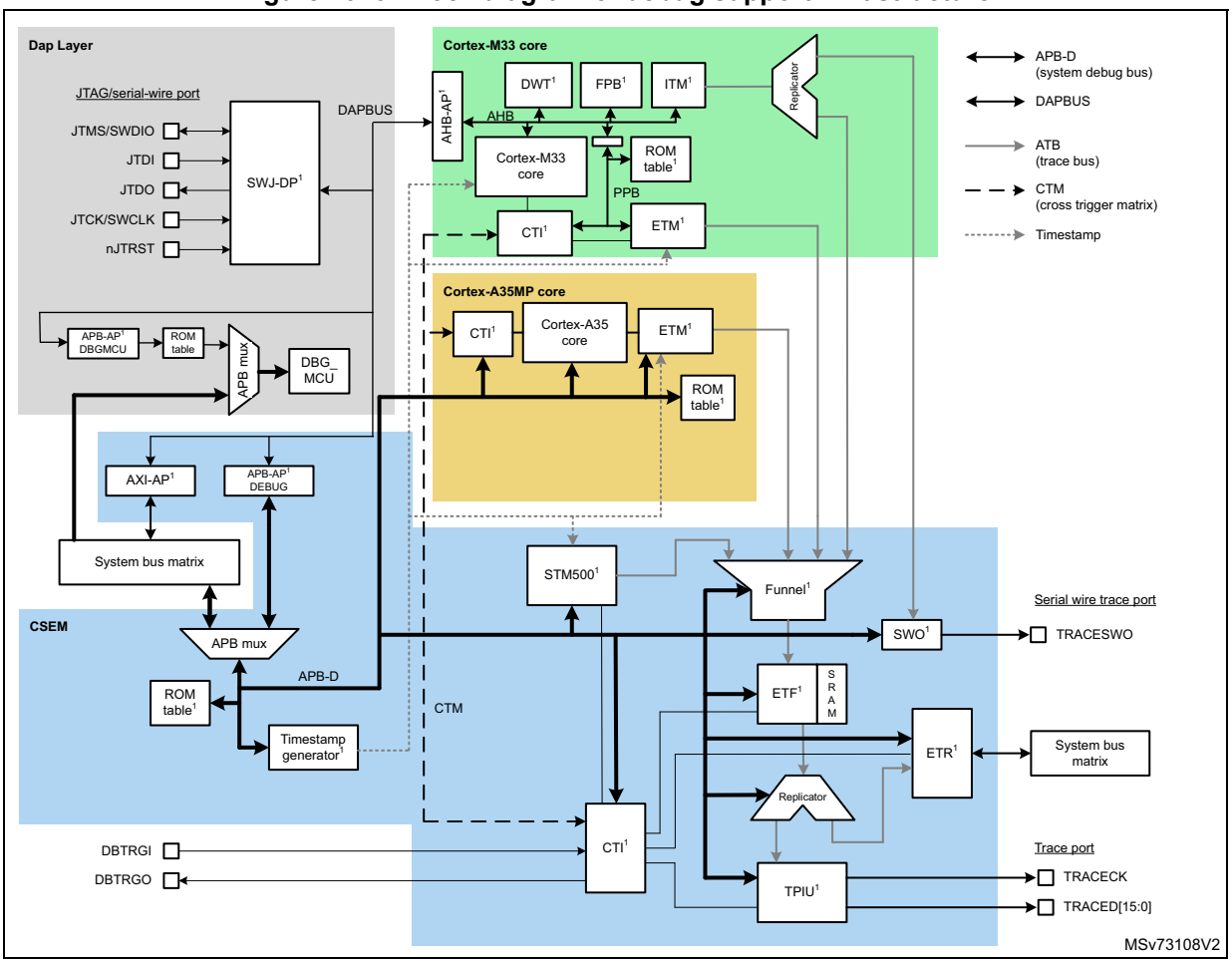
  The debugger can read the contents of the recorded trace data via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.
- Self-hosted trace analysis

  The recorded trace data can be read by any of the processor cores. This powerful feature allows built-in test software to monitor code execution in real time, analyze and identify faults, autonomously handle exceptions.
- Uploading stored trace

  The stored trace can also be uploaded to a host machine using one of the MPU communications interfaces (such as USB, USART, SPI, I2C, Ethernet, CAN). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.

80.3 DBG block diagram

Figure 1016. Block diagram of debug support infrastructure



1. Arm CoreSight component.

80.4 DBG pins

Table 894. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SWD debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	I/O	Serial-wire data in/out	-
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	-
JTDI	I	JTAG test data input	-	-	-
JTDO	O	JTAG test data output	-	-	-
NJTRST	I	JTAG test reset	-	-	-

Table 895. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED[15:0]	O	Trace synchronous data out (can be 1, 2, 4, 8 or 16 pins)	Refer to Datasheet
TRACECLK	O	Trace clock	

Table 896. Serial-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Trace asynchronous data out	(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the JTAG port in serial-wire debug interface mode, not when using JTAG.

Table 897. Trigger pins

Pin name	Type	Description	Pin assignment
DBTRGI	I	External trigger input to CTI	Refer to Datasheet
DBTRGO	O	External trigger output from CTI	

80.5 DBG power and clocking

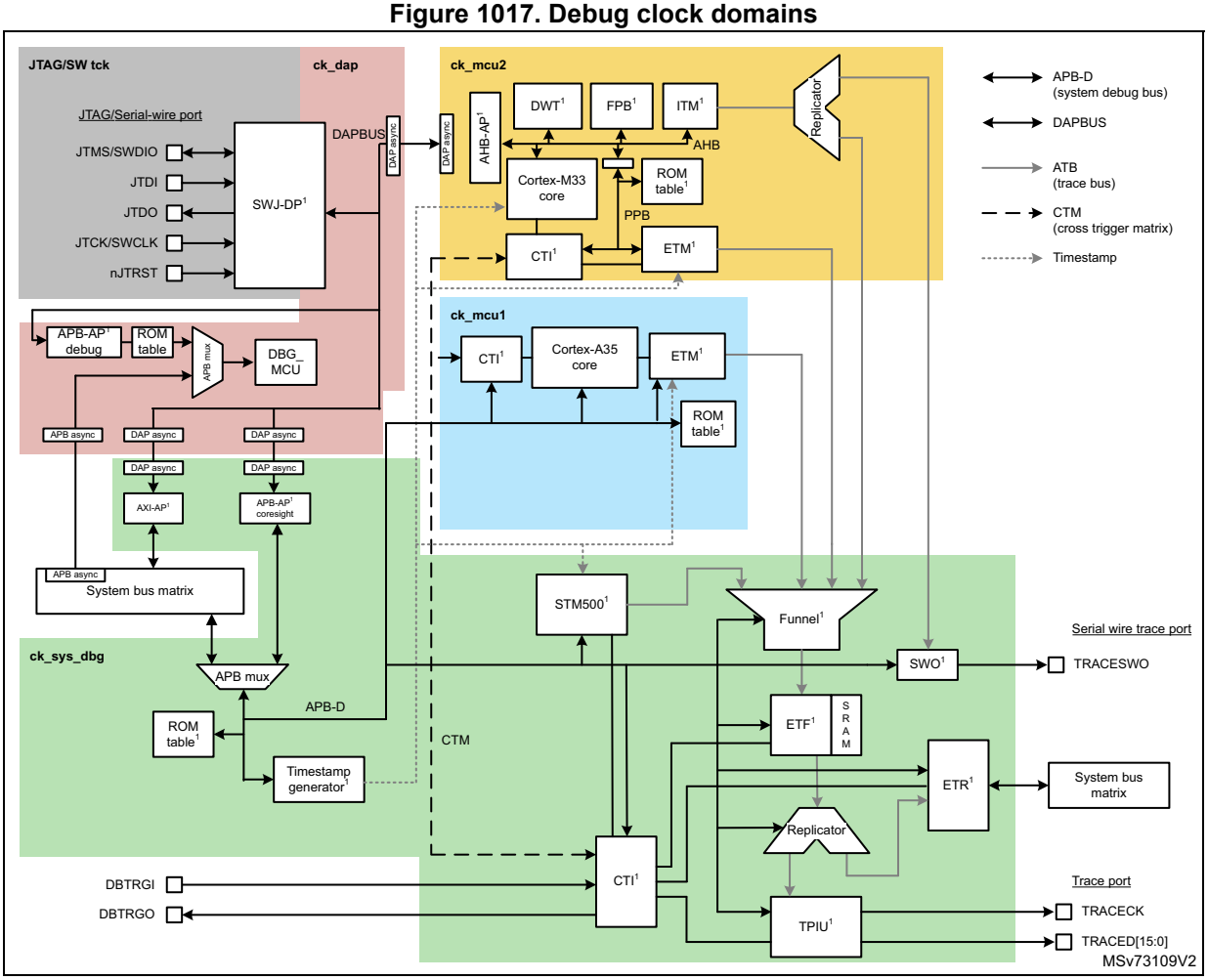
80.5.1 DBG power domains

All debug components except the SWJ-DP are in the V_{DDCORE} power domain.

The SWJ-DP is located in the V_{DD} power domain, and as such is powered in all modes but the V_{BAT} one. This means that the debugger is able to wake the device up from Standby modes.

80.5.2 DBG clocks

Figure 1017. Debug clock domains



1. Arm CoreSight component.

The debugger supplies the clock for the debug port, SWTCLK, via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the system DAP clock domain, CK_DAP. The DBGMCU is in this domain.

CK_DAP is activated with the CDBGPWRUPREQ control bit from SWJDP. When the debugger connects, it raises this control activating the clock for the DBGMCU even if the system reset is asserted. In case no debugger is connected and the CPU want to access the DBGMCU, a register inside RCC can enable this clock. The debug APB bus is clocked by CK_SYS_DBG. It is accessed from SWJ-DP through an asynchronous bridge. As CK_SYS_DBG is not enabled when system reset is asserted, the APB debug is not accessible, nor are any of the coresight components. As a consequence, the debugger discovery is not supported.

The trace subsystem (that is all components on the ATB) is in the CK_SYS_ATB trace domain.

CK_SYS_DBG and the CK_SYS_ATB clocks are enabled by the debugger using CDBGPWRUPREQ in debug port CTRL/STAT register. They can also be enabled by software writing to a register in the RCC. The clocks must be enabled before the debugger or software can access any of the debug features on the device. They must be disabled when debug and trace are not used to avoid wasting energy.

The trace port (TPIU) output clock, TRACECLK, is generated by the flexible clock generator that can be programmed to match the capability of the IO ports. The same clock is used to drive the serial wire output (SWO).

Note: TRACECLK is enabled in the RCC. It must be activated whenever tracing is required, even if neither the TPIU nor the SWO are active (that is when using the embedded trace FIFO as a software buffer). This is to avoid filling up the SWO FIFO which would then block the trace output from the ITM in the Cortex-M33.

The debug and trace components included in the processors (such as ETM, ITM, DWG, FPB, CTI) are clocked with the corresponding core clock (CK_CPU1_SS for the Cortex-A35 subsystem, CK_CPU2_SS for the Cortex-M33 subsystem).

80.5.3 Debug and low-power modes

The device includes power saving features which allow the various domain clocks and even the core power to be switched off when not required. If the core power is switched off (Standby mode), or the system debug domain is not clocked (Stop mode), all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation is implemented. If emulation is enabled, the device still enters the programmed low-power mode, but its clock and power are maintained. In other words, the domain behaves as if it is in low-power mode, but the debugger does not lose the connection.

The Emulation mode is programmed in the MCU debug (DBGMCU) unit. For more information, refer to the *Section 80.9.11: Microcontroller debug unit (DBGMCU)*.

If the device is in Stop or Standby mode without emulation, and the debugger wishes to wake the system up, it can do so by asserting the CDBGPWRUPREQ bit in debug port CTRL/STAT register. This restarts the clocks (and switches on the high power regulator if needed) but does not generate any interrupt or event to either processor core. When the wake-up has completed, CDBGPWRUPACK in the same register goes high. In this state the debugger can access the memories and peripheral registers, and trigger a wake-up of the CPU if required.

80.5.4 Debug reset

The debug components, except for the debug port, are reset by their respective clock domain resets. The debug port (SWJ-DP) is reset by a power-on reset of the always-on domain only. The access ports can be reset under debugger control, using CDBGRSTREQ/ACK in the CTRL/STAT register of the debug port.

The debugger sets CDBGRSTREQ to request a reset. It then polls CDBGRSTACK until it goes high, indicating that the reset has completed, following which it deasserts CDBGRSTREQ.

80.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled, or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of nonsecure code.

80.6.1 Authentication signals

The following authentication signals are used by a CoreSight system to determine which features are enabled or disabled:

- DEVICEEN: Controls access to debug components via the external debug port.
  0: External debugger has no access to debug components or system interconnect.
  1: External debugger has access to debug components and system interconnect. External access to secure system interconnect depends on SPIDEN state.
- DBGEN: Global enable for all debug and trace features
  0: All debug features are disabled.
  1: Debug features in nonsecure mode are enabled. Debug features in secure mode depend on SPIDEN state.
- SPIDEN: Enables debug in secure privileged mode when DBGEN = 1.
  0: Debug features are disabled in secure privileged mode. External access to secure system interconnect is disabled.
  1: Debug features are enabled in secure privileged mode. External access to secure system interconnect is enabled.
- NIDEN: Enables trace and performance monitoring (non-invasive debug).
  0: Trace generation is disabled.
  1: Trace generation in nonsecure mode is enabled. Trace generation in nonsecure mode depends on SPIDEN state.
- SPNIDEN: Enables trace and performance monitoring in secure privileged mode when NIDEN = 1.
  0: Trace generation is disabled in secure privileged mode.
  1: Trace generation is enabled in secure privileged mode.
- DBGSWEN: Enables self-hosted debug (by a software running on a processor).
  0: Software access to debug components on the APB-AP is disabled.
  1: Software access to debug components on the APB-AP is enabled.
- AP_UNLOCK: Enables all accesses to DBGMCU (while only a few are accessible if this protection is activated, refer to register description for more information).

The Cortex-A35 and Cortex-M33 have independent sets of authentication signals to allow their debug capabilities to be set individually for each core.

The table below lists which signals apply to which debug and trace components.

Table 898. Authentication signal connectivity

-	AP_UNLOCK	DEVICEEN	DBGEN	SPIDEN	NIDEN	SPNIDEN	DBGSWEN
DBGMCU APB-AP	X(1)	-	-	-	-	-	-
Debug APB-AP	-	X	-	-	-	-	-
System bus AXI-AP	-	X	-	X	-	-	-
Cortex-M33 AHB-AP	-	X	-	X(2)	-	-	-
Cortex-M33 DWT/FPB/SCS/ITM	-	-	X(2)	X(2)	X(2)	X(2)	-
Cortex-M33 ETM	-	-	X(2)	-	-	-	-
Cortex-M33 CTI	-	-	X(2)	-	X(2)	-	-
Cortex-A35 debug unit	-	-	X(3)	X(3)	X(3)	X(3)	X
Cortex-A35 ETM	-	-	X(3)	-	X(3)	-	X
Cortex-A35 CTI	-	-	X(3)	-	X(3)	-	X
STM500	-	-	X(2)(3)	X(2)(3)	X(2)(3)	X(2)(3)	X
ETF	-	-	-	-	-	-	X
ETR	-	-	X	X	-	-	X
TSGEN	-	-	-	-	-	-	X
TPIU	-	-	-	-	-	-	X

1. DBGMCU_APB_AP is open under reset, closed immediately after SYSTEM_RST is released, until AP_UNLOCKED is configured.
2. Cortex-M33 dedicated subset of authentication signals.
3. Cortex-A35 dedicated subset of authentication signals.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component section, or to the relevant Arm technical documentation.

The initial states of the signals are set at power-on reset, according to the table below. States of the signals can be modified in BSEC_DENR and AP_UNLOCK registers after boot-up by a trusted software with secure and privilege access rights.

Refer to *Section 6: Boot and security control (BSEC)*, in the debug and trace control sections, for more details on the various combinations allowed for these signals.

Table 899. Authentication signal initial states

Device state (set by OTP fuse)	Authentication signal default state	Description
Open device	All ones	All debug and trace is enabled at boot-up.
Closed device	All zeros	All debug and trace is disabled at boot-up. Secure software must enable debug as required after booting.

In addition to the functional bits of BSEC_DENR, an error detection code is transmitted by BSEC. An error detection logic is located close to the debug components with a permanent check between functional bits and error detection code. If the values do not correspond, a security tamper is generated. This tamper can be configured to generate a reset, preventing any security issue.

80.7 Chip-level TAP controller (CLTAPC)

The JTAG interface is connected to the internal scan chain of the device. It is used for testing as well as for debug. The first component in the scan chain is the chip-level TAP controller which implements a TAP state machine based on IEEE Std 1149.1-1990.
It allows access to the built-in manufacturing test features.

The CLTAPC must be bypassed in order to access the debug support functions provided by the SWJ-DP. The bypass is done by prefixing the SWJ-DP instructions with the CLTAPC bypass instruction sequence (0b11111).

The CLTAPC also contains three 16-bit registers which can be accessed using the CTRL instruction sequence (0b01000) followed by 0b1111 to bypass the SWJ-DP.

Next, the register address is shifted into the DR (LSB first). Finally, the 16-bit register content is shifted into or out of the DR, according to whether the register is read or write.

The CLTAPC registers are listed in the table below.

Table 900. CLTAPC registers

Name	Address	Description	read/write
SECSTS	0b001	Status of BSEC_DENR register	R
DBGCTL	0b010	Connected to BSEC_JTAGINR register	W
DBGSTS	0b011	Connected to BSEC_JTAGOUTR register	R

For details on BSEC register bits, refer to *Section 6.3.14: JTAG registers*.

80.8 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- 5-pin standard JTAG interface (JTAG-DP)
- 2-pin (clock + data) serial-wire debug port (SW-DP)

The two modes are mutually exclusive, since they share the same I/O pins.

By default, the JTAG-DP is selected after a system or power-on reset. The five I/O pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and NJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

...(50 or more ones)...,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,...(50 or more ones)...

JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG pins JTDI, JTDO and NJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

80.8.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE Std 1149.1-1990.

For more details on the JTAG debug protocol, refer to the Arm debug interface Architecture specification *[1]*.

80.8.2 SWD debug port

The serial-wire debug protocol uses the two following pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises the following phases:

1. packet request (8 bits) transmitted by the host
2. acknowledge response (3 bits) transmitted by the target
3. data transfer (33 bits) transmitted by the host (for a write) or target (for a read)

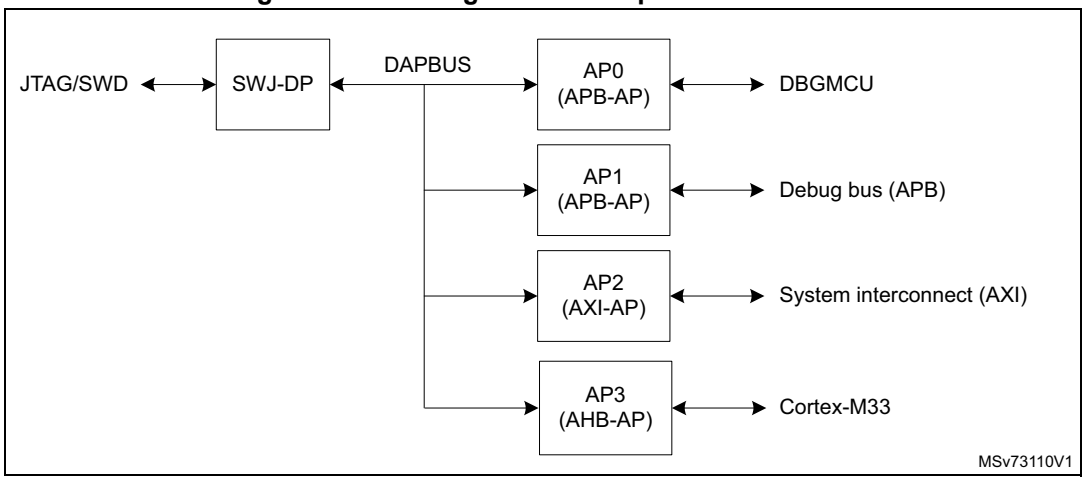
For more details on the serial wire debug protocol, refer to the Arm debug interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

80.8.3 Access ports

There are four access ports (AP) attached to the DP.

Figure 1018. Debug and access port connections



1. AP0: DBGMCU access port (APB-AP). Allows access to DBGMCU only.
2. AP1: Debug access port (APB-AP). Allows access to the CoreSight™ components on the debug APB bus, including the Cortex-A35.
3. AP2: System bus access port (AXI-AP). Allows access to the system bus matrix. This provides visibility on all the memory and peripherals accessible via the AXI and AHB interconnects, except those on the system APB debug bus. No CoreSight™ components are accessible via this port.
4. AP3: Cortex-M33 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex-M33 processor core via its PPB bus.

All four access ports are of type MEM-AP: debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus.

For more detailed information on the MEM-AP, refer to the Arm debug interface architecture specification *[1]*.

80.8.4 Authentication

The effect of the authentication signals for each access port is shown in the tables below.

Table 901. AP0 (DBGMCU APB) authentication behavior

Device state	AP_UNLOCK	CPU write once any value in authentication device register	AP status	DBGMCU register authorized registers
Power reset released, system reset asserted	0	NA	AP opened	DeviceId, DENABLE, all authentication registers accessible
System reset released, opened part	B4	Yes or no	AP opened	All registers accessible
System reset released, closed part	B4	Yes or no	AP opened	All registers accessible
System reset released, closed part	Others	No	AP closed	NA
System reset released, closed part	Others	Yes	AP opened	**Authorized registers**: DeviceId, DENABLE, all authentication registers **Locked registers**: debug control, freeze registers

Table 902. AP1 (debug APB) authentication behavior

DEVICEEN	Others authentication signals	Behavior
0	X	All access blocked.
1	-	All access enabled. *Note: Access to certain components may return an error depending on the state of the other authentication signals.*

Table 903. AP2-3 authentication behavior

DEVICEEN	SPIDEN	Others authentication signals	Behavior
0	X	X	All transfers blocked. Any attempt to perform a transfer will result in an error.
1	0	-	Secure transfers blocked. Any attempt to perform a secure transfer will result in an error. Nonsecure transfers are enabled.
1	1	-	Both nonsecure and secure transfers are enabled.

80.9 System debug features

The system debug features include the following CoreSight components:

- System ROM tables
- Global trace timestamp generator (TSGEN)
- System cross-trigger interface (CTI)
- Cross-trigger matrix (CTM)
- Trace port interface Unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Embedded trace router (ETR)
- Replicator
- System trace macrocell (STM)
- Serial wire output (SWO)

These components are accessible by the debugger via the system APB-AP and its associated APB debug bus. They are also accessible from the Cortex-A35 and Cortex-M33 processors to the extent that self-hosted feature is enabled.

The MCU debug unit (DBGMCU) is also accessed via the debug APB. This non-CoreSight component contains registers for configuring the behavior of the device in debug mode.

80.9.1 System ROM tables

There is one main ROM table on the debug APB bus. The ROM table is a CoreSight component that contains the base addresses of CoreSight debug components on APB bus. This table allows a debugger to discover automatically the CoreSight system topology.

The ROM table points to the CoreSight components located in the debug subsystem, and to a secondary ROM table in the Cortex-A35 subsystem.

The table occupies a 4-Kbyte, 32-bit wide chunk of APB address space:

- from 0x8000 0000 to 0x8000 0FFC when accessed by the JTAG debugger through the debug APB-AP
- from 0x4A00 0000 to 0x4A00 0FFC when accessed from the system bus

Each component referenced in the ROM table occupies a 64-Kbyte range.

Table 904. Debug subsystem ROM table

Address offset in ROM table	Component name	Component base address (system bus)	Component base address (debugger)	Component address offset	Data
0x000	Funnel	0x4A01 0000	0x8001 0000	0x10000	0x0001 0003
0x004	ETF	0x4A02 0000	0x8002 0000	0x20000	0x0002 0003
0x008	ETR	0x4A03 0000	0x8003 0000	0x30000	0x0003 0003
0x00C	TPIU	0x4A04 0000	0x8004 0000	0x40000	0x0004 0003
0x010	Replicator	0x4A05 0000	0x8005 0000	0x50000	0x0005 0003
0x014	Timestamp generator	0x4A06 0000	0x8006 0000	0x60000	0x0006 0003

Table 904. Debug subsystem ROM table (continued)

Address offset in ROM table	Component name	Component base address (system bus)	Component base address (debugger)	Component address offset	Data
0x018	STM500	0x4A07 0000	0x8007 0000	0x70000	0x0007 0003
0x01C	CTI0	0x4A08 0000	0x8008 0000	0x80000	0x0008 0003
0x020	CTI1	0x4A09 0000	0x8009 0000	0x90000	0x0009 0003
0x024	SWO	0x4A0A 0000	0x800A 0000	0xA0000	0x000A 0003
0x028	Cortex-A35 ROM table	0x4A20 0000	0x8020 0000	0x200000	0x0020 0003
0x02C	Top of table	-	-	-	0x0000 0000
0x030 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-

The secondary ROM table below lists the debug components attached to the Cortex-A35 processors. It occupies a 4-Kbyte, 32-bit wide chunk of APB address space:

- from 0x8020 0000 to 0x8020 0FFC when accessed by the debugger
- from 0x4A20 0000 to 0x4A20 0FFC when accessed from the system bus

Table 905. Cortex-A35 subsystem ROM table

Address offset in ROM table	Component name	Component base address (system bus)	Component base address (debugger)	Component address offset	Entry
0x000	CA35_0_DBG	0x4A21 0000	0x8021 0000	0x1 0000	0x0001 0003
0x004	CA35_0_CTI	0x4A22 0000	0x8022 0000	0x2 0000	0x0002 0003
0x008	CA35_0_PMU	0x4A23 0000	0x8023 0000	0x3 0000	0x0003 0003
0x00C	CA35_0_ETM	0x4A24 0000	0x8024 0000	0x4 0000	0x0004 0003
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-

Finally, there is a ROM table on DBGMCU APB-AP at address 0x8000 0000 that points to the DBGMCU component. This ROM table is accessible only by the debugger, while the DBGMCU is accessible by both debugger and CPU.

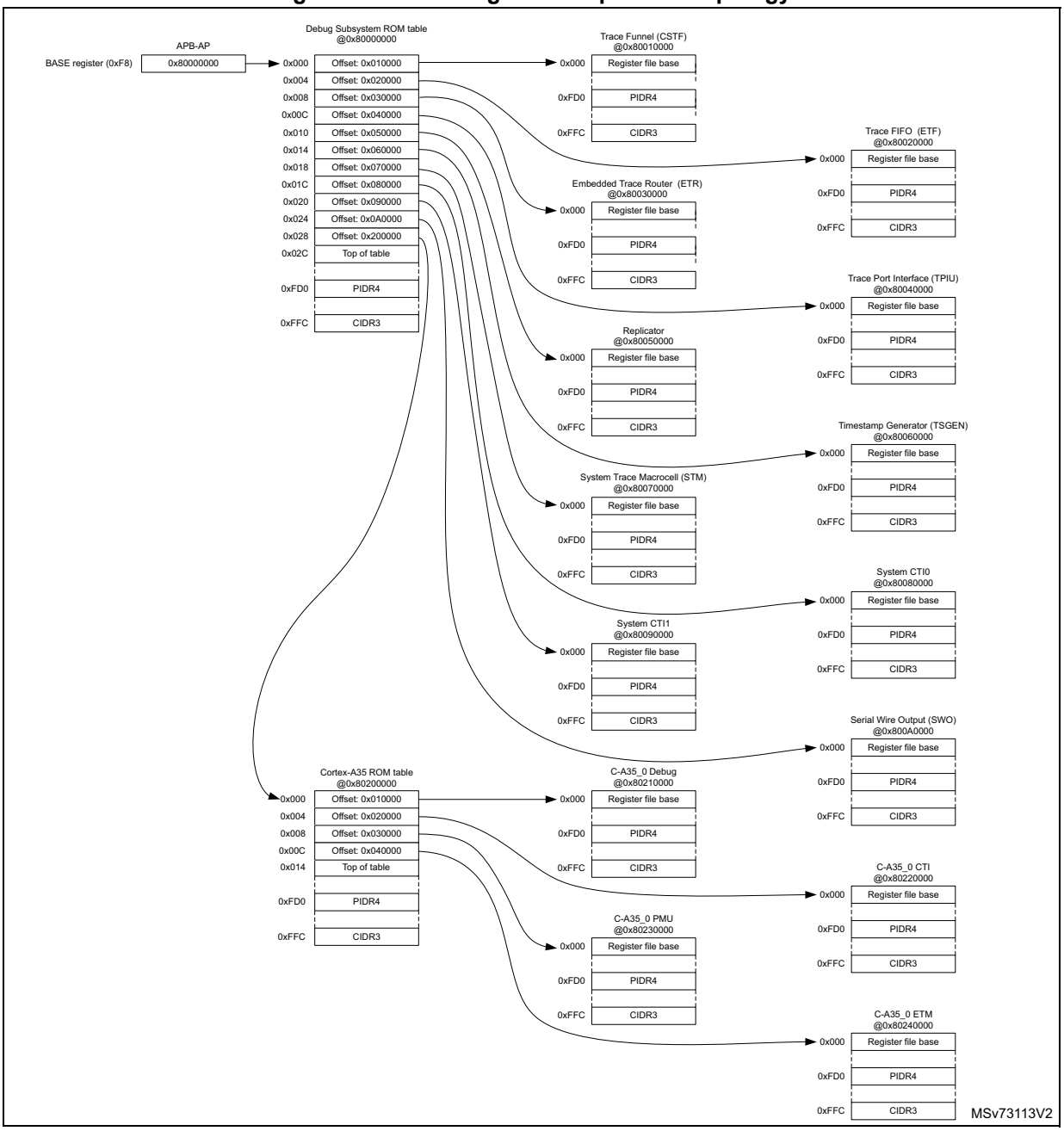
Table 906. DBGMCU ROM table

Address offset in ROM table	Component name	Component base address (system bus)	Component base address (debugger)	Component address offset	Data
0x000	DBGMCU	0x440A 0000	0x8000 1000	0x1000	0x0000 1003

For more information on these ROM tables, refer to the Arm Debug Interface Architecture Specification *[1]*.

The topology for the CoreSight™ components on the debug APB is shown in *Figure 1019*.

Figure 1019. CoreSight™ components topology



80.9.2 CoreSight global timestamp generator (TSGEN)

The CoreSight global timestamp generator contains a 64-bit counter that provides a common timing reference for all trace sources in the system (ETM, ITM and STM). These components insert timestamps in the trace streams that allow the trace analyzer to recover

the chronological order of trace packets, which can be lost when multiple trace sources are multiplexed into one stream at the funnel.

TSGEN registers are accessible over the debug APB. This allows the debugger or debug software to perform the following tasks:

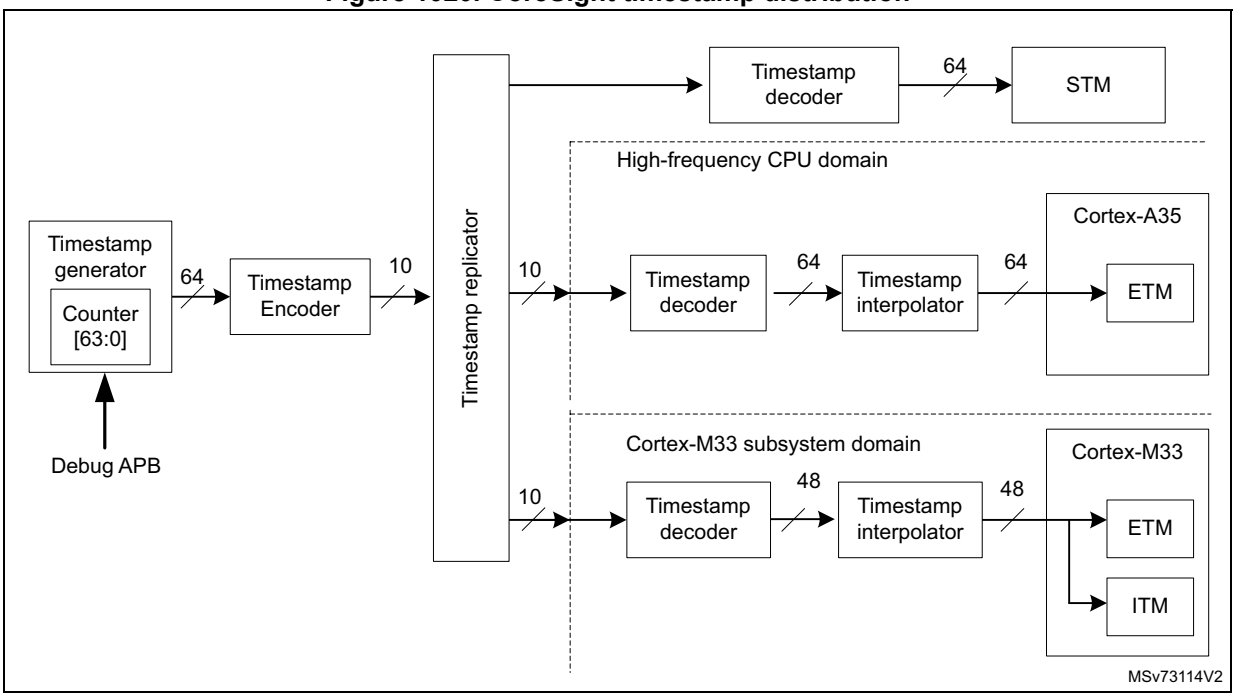
- Start and stop the timestamp incrementing.
- Cause the timestamp counter to stop when debug state is entered. To enable this feature, the debug subsystem CTI must be configured to signal to the timestamp generator when system-wide debug state has been entered (all CPU cores halted).
- Read the current timestamp value.
- Change the current timestamp value. The timestamp counter must be halted while it is changed. When the timestamp value is changed, the timestamp generator resynchronizes all trace sources.
- Change the reported timestamp increment.

For more information on the TSGEN CoreSight component, refer to the Arm CoreSight SoC-400 technical reference manual *[2]*.

The timestamp generator has its own clock domain CK_KER_TSGEN, and is distributed to the Cortex-A35 and Cortex-M33 to provide a time information for their respective trace macrocell, and to the STM to provide a time information for stamping its messages.
To simplify the distribution over clock domain boundaries, the 64-bit timestamp is encoded in 7 bits, then decoded in the destination power domain. The timestamp distribution is shown in *Figure 1020*.

Whenever the frequency of TSGEN or Cortex-M33 changes, the ETM of the Cortex-M33 processor receives a pulse on TSCLKCHANGE input.

Figure 1020. CoreSight timestamp distribution

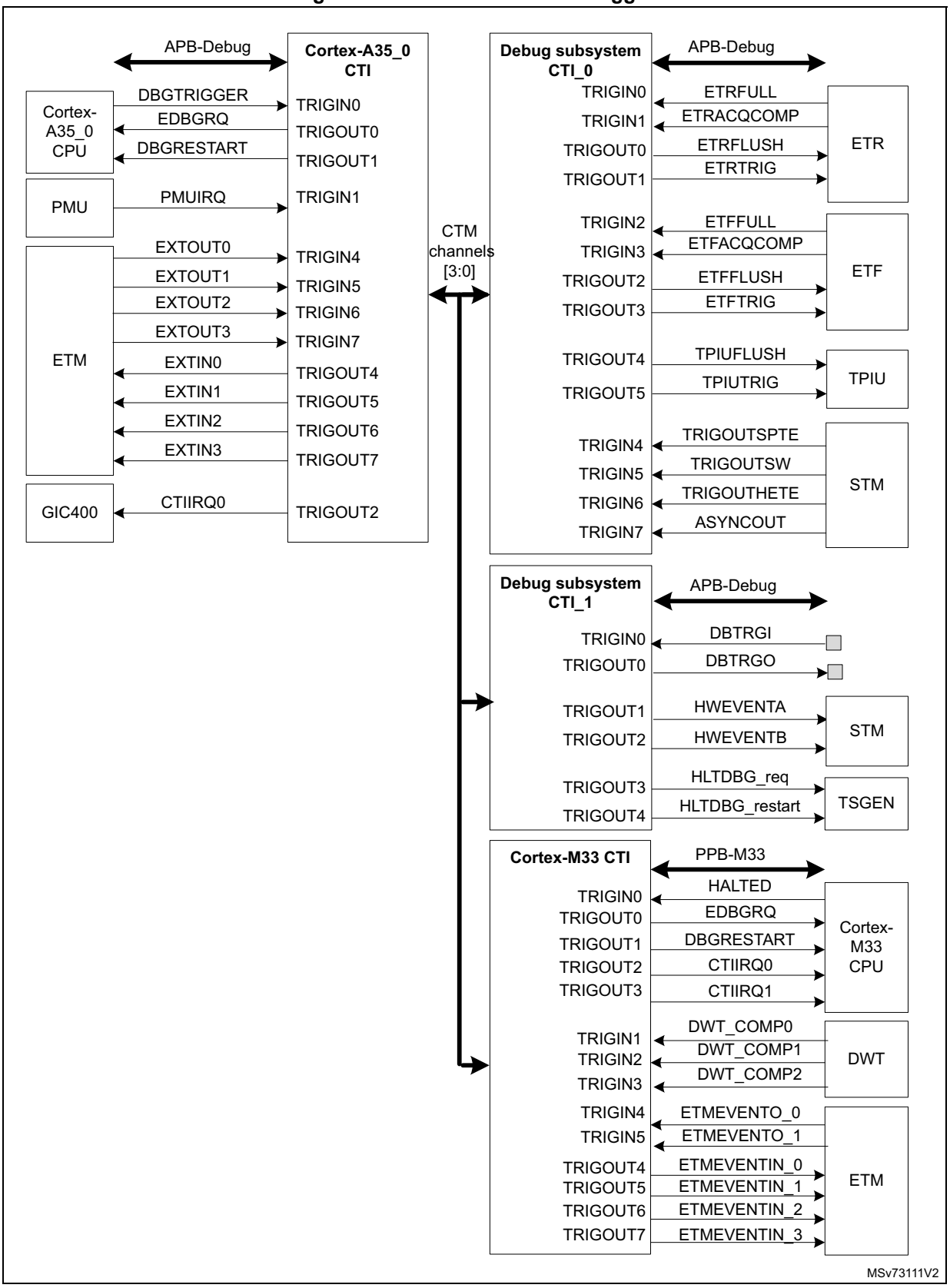


80.9.3 Cross trigger interface (CTI) and matrix (CTM)

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) form together the CoreSight™ embedded cross trigger feature. There are four CTI components, two in the debug subsystem, one for Cortex-A35 and one dedicated to the Cortex-M33.

The CTIs are connected to each other via a CTM. The debug subsystem CTIs and the Cortex-A35 CTI are accessible to the debugger via the debug APB. The Cortex-M33 CTI is accessible on its private peripheral bus (PPB) via the Cortex-M33 AHB access port.

Figure 1021. Embedded cross trigger



The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a breakpoint reached in one of the processor cores can stop the other processors, or a transition detected on an external trigger input can start code trace.

Each CTI has up to eight trigger inputs and eight trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM. The trigger input and output signals for each CTI are listed in tables below.

Table 907. Trace subsystem CTI_0 inputs

-	Source signal	Source component	Comments
0	ETRFULL	ETR	ETR full flag. Allows a debug event to be generated when the trace buffer is full.
1	ETRACQCOMP	ETR	ETR capture finished. Allows a debug event to be generated when trace acquisition is complete.
2	ETFFULL	ETF	ETF full flag. Allows a debug event to be generated when the trace FIFO is full.
3	ETFACQCOMP	ETF	ETF capture finished. Allows a debug event to be generated when the trace FIFO is empty.
4	TRIGOUTSPTE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using STMSPTER.
5	TRIGOUTSW	STM	The STM asserts this signal for one clock cycle when a trigger event is generated on writes to a TRIG location in the extended stimulus port registers.
6	TRIGOUTHETE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using STMHETER.
7	ASYNCOUT	STM	Alignment synchronization signal. The STM asserts this signal for one clock cycle when an ASYNC-VERSION-FREQ sequence is output on the ATB interface, and the ASYNCOUT signal can be used for cross-triggering.

Table 908. Trace subsystem CTI_0 outputs

-	Output signal	Destination component	Comments
0	ETRFLUSH	ETR	ETR flush trigger. Causes trace sources to be flushed into destination memory.
1	ETRTRIG	ETR	ETR trigger. Causes various kinds of flush, stops the TMC, and causes additional trigger insertion.
2	ETFFLUSH	ETF	ETF flush trigger. Causes trace FIFO to be flushed.
3	ETFTRIG	ETF	ETF enable trigger. Starts filling the trace FIFO.
4	TPIUFLUSH	TPIU	Trace port flush trigger. Causes TPIU FIFO to be flushed.
5	TPIUTRIG	TPIU	Trace port enable trigger. Starts trace output on the external trace port.
6	-	-	Not used
7	-	-	Not used

Table 909. Trace subsystem CTI_1 inputs

-	Source signal	Source component	Comments
0	DBTRGI	GPIO	External trigger input. Allows an external signal to generate a debug event
1	-	-	Not used
2	-	-	Not used
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 910. Trace subsystem CTI_1 outputs

-	Output signal	Destination component	Comments
0	DBTRGO	GPIO	External I/O trigger output. Allows monitoring of events on external DBTRGO pin.
1	HWEVENTA	STM	STM hardware event
2	HWEVENTB	STM	STM hardware event
3	HLTDBG_req	TSGEN	Halt timestamp in debug. Freezes TSGEN timestamp counter.
4	HLTDBG_restart	TSGEN	Halt timestamp in debug. Unfreezes TSGEN timestamp counter.
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 911. Cortex-A35_0 CTI inputs

-	Source signal	Source component	Comments
0	DBGTRIGGER	Cortex-A35 CPU	Pulsed high when the core enters in debug state
1	PMUIRQ	PMU	Performance monitor interrupt with inverted polarity
2	-	-	Not used
3	-	-	Not used
4	EXTOUT0	ETM	ETM external trigger output
5	EXTOUT1	ETM	ETM external trigger output
6	EXTOUT2	ETM	ETM external trigger output
7	EXTOUT3	ETM	ETM external trigger output

Table 912. Cortex-A35_0 CTI outputs

-	Output signal	Destination component	Comments
0	EDBGRQ	Cortex-A35 CPU	CPU halt request. Puts the CPU in debug mode.
1	DBGRESTART	Cortex-A35 CPU	CPU restart request. The CPU exits debug mode.
2	CTIIRQ	GIC	CTI interrupt
3	-	-	Not used
4	EXTIN0	ETM	ETM external input
5	EXTIN1	ETM	ETM external input
6	EXTIN2	ETM	ETM external input
7	EXTIN3	ETM	ETM external input

Table 913. Cortex-M33 CTI inputs

-	Source signal	Source component	Comments
0	HALTED	Cortex-M33 CPU	Pulsed high when the core enters in debug state
1	DWTCOMP_0	Cortex-M33 DWT	DWT comparator output
2	DWTCOMP_1	Cortex-M33 DWT	DWT comparator output
3	DWTCOMP_2	Cortex-M33 DWT	DWT comparator output
4	ETMEVENTO_0	Cortex-M33 ETM	ETM event output
5	ETMEVENTO_0	Cortex-M33 ETM	ETM event output
6	-	-	Not used
7	-	-	Not used

Table 914. Cortex-M33 CTI outputs

-	Source signal	Destination component	Comments
0	EDBGRQ	Cortex-M33 CPU	CPU halt request. Puts the CPU in debug mode.
1	DBGRESTART	Cortex-M33 CPU	CPU restart request. The CPU exits debug mode.
2	IRQ0	Cortex-M33 NVIC	CTI interrupt
3	IRQ1	Cortex-M33 NVIC	CTI interrupt
4	ETMEVENTIN_0	Cortex-M33 ETM	ETM event input
5	ETMEVENTIN_1	Cortex-M33 ETM	ETM event input
6	ETMEVENTIN_2	Cortex-M33 ETM	ETM event input
7	ETMEVENTIN_3	Cortex-M33 ETM	ETM event input

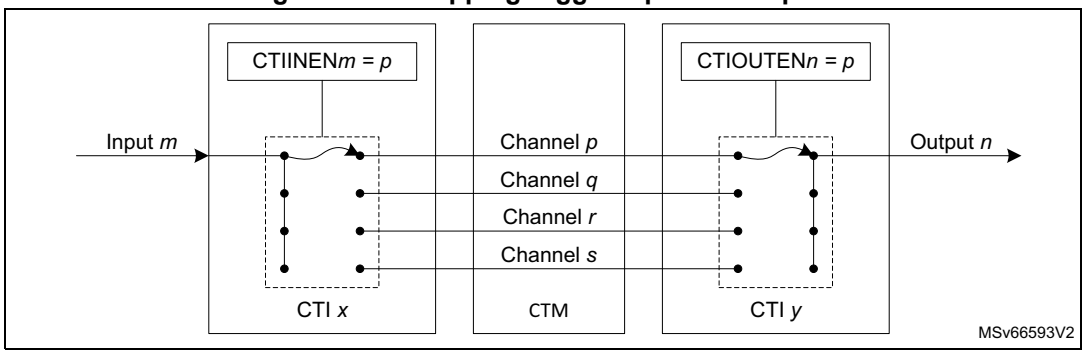
There are four event channels in a cross trigger matrix, which allows up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs.

To connect input number *m* on CTI*x* to output number *n* on CTI*y*, the input must be connected to an event channel *p* using the CTIINEN*m* register of CTI*x*. The same channel *p* must be connected to the output using the CTIOUTEN*n* register of CTI*y*.

Note: This applies even if the input and the output belong to the same CTI.

An input can be connected to more than one channel (up to four). An input can then be routed to several outputs. Similarly, an output can be connected to several inputs. Several inputs/outputs can also be connected to the same channel.

Figure 1022. Mapping trigger inputs to outputs



Example configuration: When either the MCU or the MPU hits a breakpoint, stop the other processor and restart the two processors synchronously.

To stop both cores, when either core stops requires the debug status output of each core (HALTED for Cortex-M33, DBGTRIGGER for Cortex-A35) to be connected to the EDBGRQ input of the opposite core.

As mentioned in *Table 911* and *Table 913*, the DBGTRIGGER signal from Cortex-A35_0 processor core0 is connected to input 0 of the Cortex-A35_0 CTI. The same signal from the Cortex-M33 core is connected to the same input on the Cortex-M33 CTI. The programming of the CTIIEN0 register on each CTI connects these inputs to a CTM channel (for example channel 0).

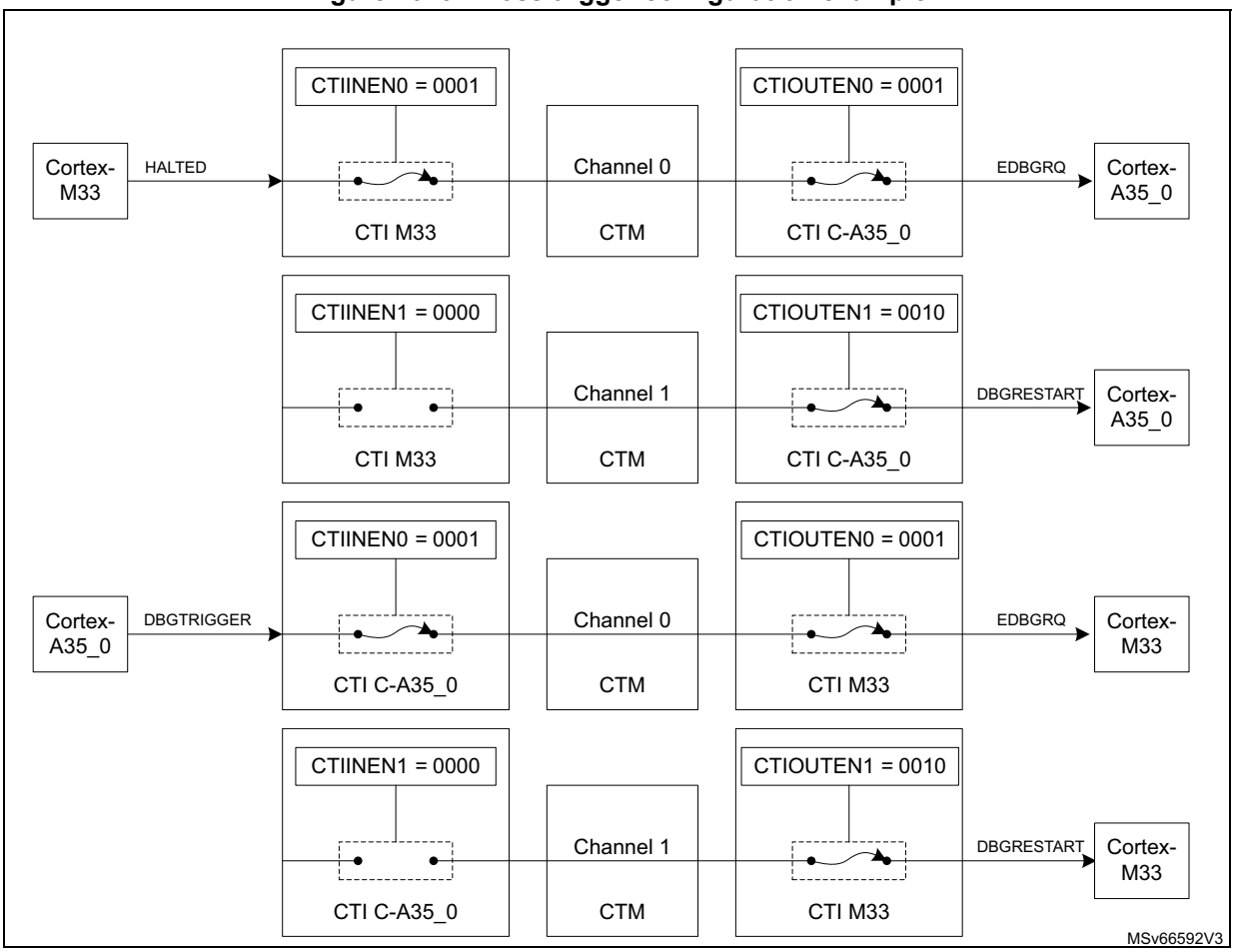
As mentioned in *Table 912* and *Table 914*, the EDBGRQ signals to the CPUs are connected to output 0 of their respective CTIs. The programming of the CTIIEN0 register on each CTI connects these outputs to the same CTM channel.

To restart both cores simultaneously the debugger must use the APPPULSE register in either of the CTIs. This allows the debugger to generate a pulse on any of the four CTM channels. The channel must be connected to the DBGRESTART signal of both cores.

As mentioned in *Table 912* and *Table 914*, the DBGRESTART signals to the CPUs are connected to output 1 of their respective CTIs. The programming of the CTIOUTEN1 register on each CTI connects these outputs to another CTM channel (for example, channel 1).

The above configuration is illustrated in the figure below.

Figure 1023. Cross trigger configuration example



To force the processors to restart simultaneously, the debugger must use the following procedure:

1. Clear debug request by writing 0x01, then 0x00, to CTIINTACK register in each CTI.
2. Cause a pulse on channel 1 by writing 0x02 to APPPULSE register in either CTI. This generates a restart request to both processors.

Note: The debugger can also force both cores to stop simultaneously by writing 0x01 to the APPPULSE register in either CTI, which generates a pulse on channel 0.

Cross trigger interface registers

The register base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

For more information on the CTI CoreSight component, refer to the Arm CoreSight SoC-400 technical reference manual *[2]*.

80.9.4 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines the ATB buses from the five trace sources into one single ATB. The CSTF has five ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: Cortex-A35_0 ETM
- S1: Reserved
- S2: STM
- S3: Cortex-M33 ETM
- S4: Cortex-M33 ITM

The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when trace is disabled. The arbitration works as follows:

1. The arbiter selects the slave port with the highest assigned priority that has data valid.
2. Up to min_hold_time transfers are passed from the selected slave to the master port, where min_hold_time is programmable in the CONTROL register.
3. A new arbitration is then performed.

A high priority must be assigned to slave ports connected to sources with a small amount of buffering, or where data loss cannot be tolerated. Low priority must be assigned to less critical sources or those with large buffers.

For more information on the CSTF CoreSight component, refer to the Arm CoreSight SoC-400 technical reference manual *[2]*.

80.9.5 Embedded trace FIFO (ETF)

The ETF is a 64-Kbyte memory which captures trace data from five trace sources:

- ETM of Cortex-A35 core
- ITM and ETM of Cortex-M33
- STM

The ETF is a configuration of the CoreSight trace memory controller component. It can be used in the following modes (selected in the mode register):

- Hardware FIFO mode

  The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled. When the trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.

  In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty in nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very big.

- Software FIFO mode

  The trace memory is used as a FIFO that can be read through the RRD register while trace is being captured. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled.

  This mode allows the trace to be transferred by DMA into the system memory, or to a high-speed interface (such as SPI, USB), or even monitored by software running on one of the cores.

Note: Unlike the hardware FIFO mode, this mode is invasive, since it uses system resources which are shared by the processors.

- Circular buffer mode (embedded trace buffer)

  The trace memory is used as a circular buffer. Trace data is captured into the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition occurs.

  In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to the ETF SRAM one (64 Kbytes in this case). Being a circular buffer, if the FIFO becomes full, incoming trace data overwrites the oldest stored data, which are lost. Therefore the contents of the trace buffer represent the most recent activity of the processor(s), up to the point when the buffer was stopped, rather than all the activity since the trace was started.

  The possible methods to read out the buffer contents once the trace has stopped are details below:

  – via the trace port: With the TPIU enabled, the contents of the buffer are output over the trace port. This can be done by setting the DrainBuffer bit in the ETF_FFCR register.
  – via the ETR: The buffer can be sent out to memory, either internal such as SRAM, or external such as DDR. With the ETR enabled, this can be done by setting the DrainBuffer bit in the ETF_FFCR register.
  – via the debug port: the debugger can read the buffer via the RRD register which is accessible over the debug APB.
  – by software: Either processor can read the buffer via the RRD register, since the debug APB is accessible from the system bus. This can also be performed by software triggered DMA transfers. No hardware DMA request is available so the FULL flag can be used to trigger an interrupt to the appropriate core via the cross trigger. The software can then read the ETF_CBUFLVL register to find out the number of data to be read.

For more information on the trace memory controller CoreSight component, refer to the Arm CoreSight Trace Memory Controller technical reference manual *[3]*.

80.9.6 Embedded trace router (ETR)

The ETR allows trace data from the Arm trace bus to be sent to external memory, using a DMA scatter/gather interface over an AXI bus.

The ETR also supports programmable burst options for an optimum efficiency of the system interconnect and memory controller. The trace capture can be intrusive to system performance, because less memory is available to the rest of the system, and because the bandwidth must be shared between the ETR and regular system functions. However, the depth of trace capture available is likely to be much larger than available when storing in the

ETF and a trace port is not required. Reading of the AXI based trace buffer can either be done directly over the AXI from a normal bus master or through the ETR as if it is an ETB.

The ETR is a configuration of the CoreSight trace memory controller component. It is configured at design time, and some registers accessible through the APB interface allows some parameters to be modified. In particular, the following:

- AXI control register: It contains a field indicating the maximum number of data transfers that can occur within each burst initiated by the TMC on the AXI master interface.
- data buffer address low and high registers: They contain the 32-bit address used to locate the trace buffer in the system memory.
- formatter and flush control register: It enables the trace data to be TWP formatted with only frames (no padding or syncs). When disabled, the data is raw data from the ATB input, without the ID.

A low-power interface is implemented in order to allow the debugger to reset the ETR without provoking protocol violation on the global AXI bus interconnect. The LPI interface is controlled from a configuration register located in the Cortex-A35 system configuration.

From a security perspective, the ETR is associated with a RIMU that controls its master access to the interconnect (see *Section 8: Resource isolation framework security controller (RIFSC)* for more details on the RIMU).

80.9.7 Trace port interface unit (TPIU)

The TPIU is a CoreSight component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and sixteen data outputs, TRACED(15:0). The trace port width is programmable in the range 1 to 16. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes. It also restricts the bandwidth of the trace port and the quantity of trace information that can be output in real time.

For more information on TPIU CoreSight component, refer to the Arm CoreSight SoC-400 technical reference manual *[2]*.

80.9.8 Replicator

This Arm CoreSight component replicates an Arm trace bus input to two identical Arm trace bus outputs. The replicator does not accept more data from the trace source until all trace sinks have accepted this data. This reduces the throughput of the replicator to match that of the slowest trace sink. To prevent this, the APB interface is implemented on this replicator, in order to disable paths. This gives also some filtering capabilities (for example some trace off-chip, some to the ETR).

Table 915. Replicator allocation

ATB output port	Trace sink
#0	ETR
#1	TPIU

80.9.9 Serial-wire output (SWO)

The SWO is a CoreSight component that formats the trace stream from the Cortex-M33 processor ITM, and outputs it on the single wire TRACESWO output.

Compared to the TPIU, the SWO contains:

- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output (no TRACEDx or TRACECLK pin)
- no support for flush (because this is not required)
- no support for triggering

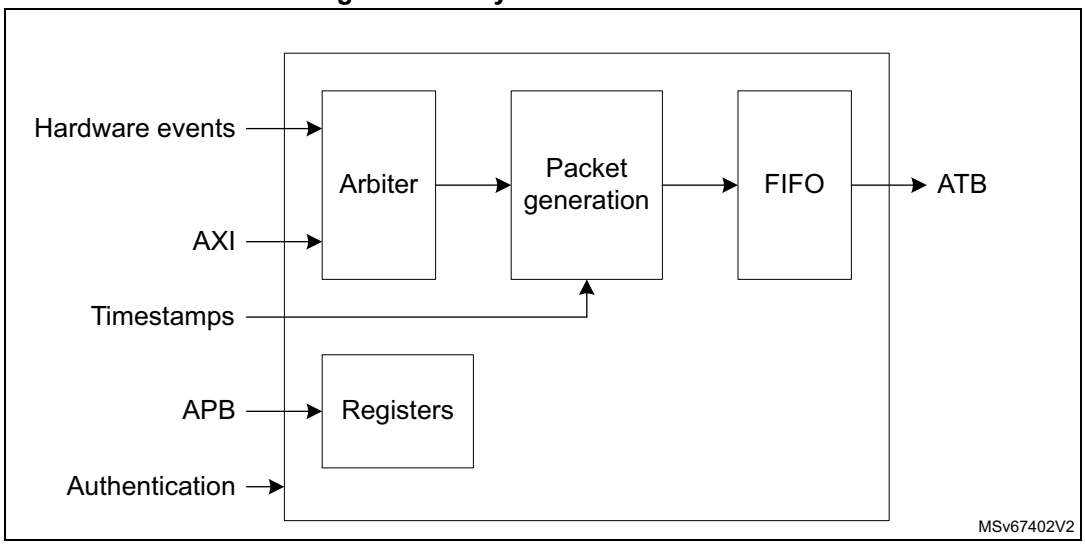
The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the SWO CoreSight component, refer to the Arm CoreSight Components technical reference manual *[4]*.

80.9.10 System trace macrocell (STM)

The system trace macrocell is a CoreSight component which allows instrumentation data generated by software to be incorporated in the trace output. The software writes the instrumentation data to the STM via a memory-mapped port on the AXI system interconnect. The data is formatted and time-stamped for output on the ATB trace bus, where it is merged with the trace coming from other sources (ETM, ITM).

Figure 1024. System trace macrocell



The STM is configured with the following values:

- DATA_FIFO_DEPTH = 16
- CHN_FIFO_DEPTH = 8
- HW_EVOBIF_PRESENT = TRUE

The various AXI masters can generate trace by writing to the extended STM stimulus ports via its AXI slave port, which is mapped in a 4-Mbyte window of address space

(see *Section 2.4: Memory organization*). A 256-byte extended stimulus port comprises 16 x 64-bit locations or registers. Writes to different locations generate different types of trace packets, as shown in the table below.

The base address for stimulus port n is the STM base address + n x 256. A master can use any number of the 16384 available stimulus ports mapped in the 4-Mbyte window.

Table 916. STM extended stimulus port memory map

Address offset	Name	Description
0x00	G_DMTS	Data, marked with timestamp, guaranteed
0x08	G_DM	Data, marked, guaranteed
0x10	G_DTS	Data, with timestamp, guaranteed
0x18	G_D	Data, guaranteed
0x20-0x58	-	Reserved
0x60	G_FLAGTS	Flag with timestamp, guaranteed
0x68	G_FLAG	Flag, guaranteed
0x70	G_TRIGTS	Trigger with timestamp, guaranteed
0x78	G_TRIG	Trigger, guaranteed
0x80	I_DMTS	Data, marked with timestamp, invariant timing
0x88	I_DM	Data, marked, invariant timing
0x90	I_DTS	Data, with timestamp, invariant timing
0x98	I_D	Data, invariant timing
0xA0-0xD8	-	Reserved
0xE0	I_FLAGTS	Flag with timestamp, invariant timing
0xE8	I_FLAG	Flag, invariant timing
0xF0	I_TRIGTS	Trigger with timestamp, invariant timing
0xF8	I_TRIG	Trigger, invariant timing

The trace packets contain the 8-bit identity of the master that triggered them. This identity is decoded from the nonsecure master ID carried on the AWID wires of the AXI interconnect. The ID also includes the security status of the master at the time.

The ID is mapped as follows:

- Bit [7] = 0
- Bit [6] = secure/nonsecure state
- Bit [5:0] = master ID

A list of valid IDs and how they map to the originating master is given in the table below.

Table 917. STM trace packet ID mapping to AXI masters

Master	Secure access ID	Nonsecure access ID
Cortex-A35 processor core0	0x00	0x40
Cortex-M33	0x02	0x42

Table 917. STM trace packet ID mapping to AXI masters (continued)

Master	Secure access ID	Nonsecure access ID
HPDMA1	0x03	0x43
HPDMA2	0x04	0x44
HPDMA3	0x05	0x45
DAP-AXI-AP	0x06	0x46

The STM can also generate trace packets or triggers when certain hardware events are detected. A maximum of 64 events can be traced at any one time. An external multiplexer allows four sets of signals to be mapped onto these events. The mapping is given in *Table 918* to *Table 920*. All events are edge sensitive.

As shown in *Table 898*, the STM is subject to the authentication signals for Cortex-A35 and Cortex-M33. This means that only a processor that has the rights to produce trace (established from BSEC_DENR) can use the STM.

For more information on the STM, refer to the Arm CoreSight STM technical reference manual *[8]*.

Table 918. Hardware event mapping 0 to 3

Bank	Event	EXTMUX(7:0) = 0	EXTMUX(7:0) = 1	EXTMUX(7:0) = 2	EXTMUX(7:0) = 3
HEBS = 0	0	HPDMA1_ch0_cpu1_irq	HPDMA2_ch0_cpu1_irq	HPDMA3_ch0_cpu1_irq	-
	1	HPDMA1_ch1_cpu1_irq	HPDMA2_ch1_cpu1_irq	HPDMA3_ch1_cpu1_irq	-
	2	HPDMA1_ch2_cpu1_irq	HPDMA2_ch2_cpu1_irq	HPDMA3_ch2_cpu1_irq	-
	3	HPDMA1_ch3_cpu1_irq	HPDMA2_ch3_cpu1_irq	HPDMA3_ch3_cpu1_irq	-
	4	HPDMA1_ch4_cpu1_irq	HPDMA2_ch4_cpu1_irq	HPDMA3_ch4_cpu1_irq	-
	5	HPDMA1_ch5_cpu1_irq	HPDMA2_ch5_cpu1_irq	HPDMA3_ch5_cpu1_irq	-
	6	HPDMA1_ch6_cpu1_irq	HPDMA2_ch6_cpu1_irq	HPDMA3_ch6_cpu1_irq	-
	7	HPDMA1_ch7_cpu1_irq	HPDMA2_ch7_cpu1_irq	HPDMA3_ch7_cpu1_irq	-
	8	HPDMA1_ch8_cpu1_irq	HPDMA2_ch8_cpu1_irq	HPDMA3_ch8_cpu1_irq	-
	9	HPDMA1_ch9_cpu1_irq	HPDMA2_ch9_cpu1_irq	HPDMA3_ch9_cpu1_irq	-
	10	HPDMA1_ch10_cpu1_irq	HPDMA2_ch10_cpu1_irq	HPDMA3_ch10_cpu1_irq	-
	11	HPDMA1_ch11_cpu1_irq	HPDMA2_ch11_cpu1_irq	HPDMA3_ch11_cpu1_irq	-
	12	HPDMA1_ch12_cpu1_irq	HPDMA2_ch12_cpu1_irq	HPDMA3_ch12_cpu1_irq	-
	13	HPDMA1_ch13_cpu1_irq	HPDMA2_ch13_cpu1_irq	HPDMA3_ch13_cpu1_irq	LPTIMER1_IT_irq
	14	HPDMA1_ch14_cpu1_irq	HPDMA2_ch14_cpu1_irq	HPDMA3_ch14_cpu1_irq	LPTIMER2_IT_irq

Table 918. Hardware event mapping 0 to 3 (continued)

Bank	Event	EXTMUX(7:0) = 0	EXTMUX(7:0) = 1	EXTMUX(7:0) = 2	EXTMUX(7:0) = 3
HEBS = 0	15	HPDMA1_ch15_cpu1_irq	HPDMA2_ch15_cpu1_irq	HPDMA3_ch15_cpu1_irq	LPTIMER3_IT_irq
	16	HPDMA1_ch0_cpu2_irq	HPDMA2_ch0_cpu2_irq	HPDMA3_ch0_cpu2_irq	LPTIMER4_IT_irq
	17	HPDMA1_ch1_cpu2_irq	HPDMA2_ch1_cpu2_irq	HPDMA3_ch1_cpu2_irq	LPTIMER5_IT_irq
	18	HPDMA1_ch2_cpu2_irq	HPDMA2_ch2_cpu2_irq	HPDMA3_ch2_cpu2_irq	-
	19	HPDMA1_ch3_cpu2_irq	HPDMA2_ch3_cpu2_irq	HPDMA3_ch3_cpu2_irq	SPI1_irq
	20	HPDMA1_ch4_cpu2_irq	HPDMA2_ch4_cpu2_irq	HPDMA3_ch4_cpu2_irq	SPI2_irq
	21	HPDMA1_ch5_cpu2_irq	HPDMA2_ch5_cpu2_irq	HPDMA3_ch5_cpu2_irq	SPI3_irq
	22	HPDMA1_ch6_cpu2_irq	HPDMA2_ch6_cpu2_irq	HPDMA3_ch6_cpu2_irq	SPI4_irq
	23	HPDMA1_ch7_cpu2_irq	HPDMA2_ch7_cpu2_irq	HPDMA3_ch7_cpu2_irq	SPI5_irq
	24	HPDMA1_ch8_cpu2_irq	HPDMA2_ch8_cpu2_irq	HPDMA3_ch8_cpu2_irq	SPI6_irq
	25	HPDMA1_ch9_cpu2_irq	HPDMA2_ch9_cpu2_irq	HPDMA3_ch9_cpu2_irq	SPI7_irq
	26	HPDMA1_ch10_cpu2_irq	HPDMA2_ch10_cpu2_irq	HPDMA3_ch10_cpu2_irq	-
	27	HPDMA1_ch11_cpu2_irq	HPDMA2_ch11_cpu2_irq	HPDMA3_ch11_cpu2_irq	-
	28	HPDMA1_ch12_cpu2_irq	HPDMA2_ch12_cpu2_irq	HPDMA3_ch12_cpu2_irq	SAI1_irq
	29	HPDMA1_ch13_cpu2_irq	HPDMA2_ch13_cpu2_irq	HPDMA3_ch13_cpu2_irq	SAI2_irq
	30	HPDMA1_ch14_cpu2_irq	HPDMA2_ch14_cpu2_irq	HPDMA3_ch14_cpu2_irq	SAI3_irq
	31	HPDMA1_ch15_cpu2_irq	HPDMA2_ch15_cpu2_irq	HPDMA3_ch15_cpu2_irq	SAI4_irq
HEBS = 1	0	EXTI2_EXTI0_irq	PWRCTL_wkup1_irq	TIM1_brk_irq	I2C1_glb_irq
	1	EXTI2_EXTI1_irq	PWRCTL_wkup2_irq	TIM1_cc_irq	I2C2_glb_irq
	2	EXTI2_EXTI2_irq	PWRCTL_wkup3_irq	TIM1_trg_irq	I2C3_glb_irq
	3	EXTI2_EXTI3_irq	PWRCTL_wkup4_irq	TIM1_upd_irq	-
	4	EXTI2_EXTI4_irq	PWRCTL_wkup5_irq	TIM2_irq	-
	5	EXTI2_EXTI5_irq	PWRCTL_wkup6_irq	TIM3_irq	-
	6	EXTI2_EXTI6_irq	-	TIM4_irq	-
	7	EXTI2_EXTI7_irq	ADC1_irq	TIM5_gbl_irq	-
	8	EXTI2_EXTI8_irq	ADC2_irq	TIM6_gbl_irq	I3C1_glb_irq
	9	EXTI2_EXTI9_irq	-	TIM7_gbl_irq	I3C2_glb_irq
	10	EXTI2_EXTI10_irq	-	TIM8_brk_irq	I3C3_glb_irq
	11	EXTI2_EXTI11_irq	FMC_glb_irq	TIM8_cc_irq	-
	12	EXTI2_EXTI12_irq	OTFDEC1_irq	TIM8_trg_irq	-
	13	EXTI2_EXTI13_irq	-	TIM8_upd_irq	SPDIF_RX_irq
	14	EXTI2_EXTI14_irq	-	TIM10_gbl_irq	SDMMC1_irq
	15	EXTI2_EXTI15_irq	OCTOSPI1_irq	TIM11_gbl_irq	SDMMC2_irq
	16	EXTI1_EXTI0_irq	-	TIM12_gbl_irq	SDMMC3_irq

Table 918. Hardware event mapping 0 to 3 (continued)

Bank	Event	EXTMUX(7:0) = 0	EXTMUX(7:0) = 1	EXTMUX(7:0) = 2	EXTMUX(7:0) = 3
HEBS = 1	17	EXTI1_EXTI1_irq	FDCAN_cal_irq	TIM13_gbl_irq	-
	18	EXTI1_EXTI2_irq	FDCAN1_intr0_irq	TIM14_gbl_irq	-
	19	EXTI1_EXTI3_irq	FDCAN1_intr1_irq	TIM15_gbl_irq	-
	20	EXTI1_EXTI4_irq	FDCAN2_intr0_irq	TIM16_gbl_irq	-
	21	EXTI1_EXTI5_irq	FDCAN2_intr1_irq	TIM17_gbl_irq	-
	22	EXTI1_EXTI6_irq	FDCAN3_intr0_irq	-	-
	23	EXTI1_EXTI7_irq	FDCAN3_intr1_irq	-	-
	24	EXTI1_EXTI8_irq	-	-	EXTI_wwdg1_wkup
	25	EXTI1_EXTI9_irq	FPU_irq	-	IWDG1_ew_irq
	26	EXTI1_EXTI10_irq	-	-	IWDG2_ew_irq
	27	EXTI1_EXTI11_irq	-	-	IWDG3_ew_irq
	28	EXTI1_EXTI12_irq	LTDC_cpu1_l01_irq	-	IWDG4_ew_irq
	29	EXTI1_EXTI13_irq	LTDC_cpu1_l01_err_irq	-	-
	30	EXTI1_EXTI14_irq	LTDC_cpu1_l2_irq	-	WWDG1_ew_irq
	31	EXTI1_EXTI15_irq	LTDC_cpu1_l2_err_irq	-	-

Table 919. Hardware event mapping 4 to 7

Bank	Event	EXTMUX(7:0) = 4	EXTMUX(7:0) = 5	EXTMUX(7:0) = 6	EXTMUX(7:0) = 7
HEBS = 0	0	HWEVENTA -	LTDC_cpu2_l01_irq	MDF1_FLT0_irq	USART1_irq
	1	HWEVENTA ¯	LTDC_cpu2_l01_err_irq	MDF1_FLT1_irq	USART2_irq
	2	HWEVENTB -	LTDC_cpu2_l2_irq	MDF1_FLT2_irq	USART3_irq
	3	HWEVENTB ¯	LTDC_cpu2_l2_err_irq	MDF1_FLT3_irq	UART4_irq
	4	-	-	MDF1_FLT4_irq	UART5_irq
	5	cpu2_sysreset_irq	DCMIPP_glb_irq	MDF1_FLT5_irq	USART6_irq
	6	VDDcpu_vd_irq	-	MDF1_FLT6_irq	UART7_irq
	7	VDDcore_vd_irq	-	MDF1_FLT7_irq	-
	8	TEMP_irq	-	-	-
	9	RETRAMcrc_err_irq	CSIhost_irq	-	LPUART1_irq
	10	-	CCI_irq	IPCC1_rx0_cpu1_irq	-
	11	CRYP1_irq	-	IPCC1_rx0_cpu1_s_irq	USART1_RX_sreq
	12	CRYP2_irq	ETH1_lpi_irq	IPCC1_tx0_cpu1_irq	USART1_TX_sreq
	13	SAES_irq	ETH1_pmt_irq	IPCC1_tx0_cpu1_s_irq	USART2_RX_sreq
	14	HASH_irq	ETH1_sbd_irq	-	USART2_TX_sreq

Table 919. Hardware event mapping 4 to 7 (continued)

Bank	Event	EXTMUX(7:0) = 4	EXTMUX(7:0) = 5	EXTMUX(7:0) = 6	EXTMUX(7:0) = 7
HEBS = 0	15	PKA_irq	ETH2_lpi_irq	-	USART3_RX_sreq
	16	RNG_irq	ETH2_pmt_irq	-	USART3_TX_sreq
	17	TAMP_cpu1_s_wkup	ETH2_sbd_irq	-	UART4_RX_sreq
	18	TAMP_cpu1_wkup	-	IPCC1_rx1_cpu2_irq	UART4_TX_sreq
	19	TAMP_cpu2_s_wkup	ETHSW_fsc_irq	IPCC1_rx1_cpu2_s_irq	UART5_RX_sreq
	20	TAMP_cpu2_wkup	ETHSW_msg_buf_irq	IPCC1_tx1_cpu2_irq	UART5_TX_sreq
	21	-	ETHSW_combined_irq	IPCC1_tx1_cpu2_s_irq	USART6_RX_sreq
	22	-	-	-	USART6_TX_sreq
	23	IAC_cpu1_irq	-	-	UART7_RX_sreq
	24	IAC_cpu2_irq	-	-	UART7_TX_sreq
	25	-	-	-	-
	26	RTC_cpu1_wkup	rcc_hsifmon_cpu1_irq	-	-
	27	RTC_cpu1_s_wkup	rcc_hsifmon_cpu2_irq	-	-
	28	RTC_cpu2_wkup	-	ddrphy_irq	-
	29	RTC_cpu2_s_wkup	TIM8_CH1_sreq	ddrperfm_irq	LPUART1_RX_sreq
	30	-	TIM8_CH2_sreq	umctl2_irq	LPUART1_TX_sreq
	31	-	-	-	-
HEBS = 1	0	TIM1_CH1_sreq	TIM8_CH3_sreq	dfi_error_irq	SPI1_RX_sreq
	1	TIM1_CH2_sreq	TIM8_CH4_sreq	SERF_irq	SPI1_TX_sreq
	2	TIM1_CH3_sreq	TIM8_UP_sreq	-	SPI2_RX_sreq
	3	TIM1_CH4_sreq	TIM8_TRIG_sreq	I2C1_EVC_sreq	SPI2_TX_sreq
	4	TIM1_TRIG_sreq	TIM8_COM_sreq	I2C2_EVC_sreq	SPI3_RX_sreq
	5	TIM1_UP_sreq	TIM15_CH1_sreq	I2C3_EVC_sreq	SPI3_TX_sreq
	6	TIM1_COM_sreq	TIM15_UP_sreq	-	SPI4_RX_sreq
	7	TIM2_CH1_sreq	TIM15_TRIG_sreq	-	SPI4_TX_sreq
	8	TIM2_CH2_sreq	TIM15_COM_sreq	-	SPI5_RX_sreq
	9	TIM2_CH3_sreq	TIM16_CH1_sreq	-	SPI5_TX_sreq
	10	TIM2_CH4_sreq	TIM16_UP_sreq	-	SPI6_RX_sreq
	11	TIM2_UP_sreq	TIM17_CH1_sreq	I2C1_RX_sreq	SPI6_TX_sreq
	12	TIM3_CH1_sreq	TIM17_UP_sreq	I2C1_TX_sreq	SPI7_RX_sreq
	13	TIM3_CH2_sreq	-	I2C2_RX_sreq	SPI7_TX_sreq
	14	TIM3_CH3_sreq	-	I2C2_TX_sreq	-
	15	TIM3_CH4_sreq	-	I2C3_RX_sreq	-
	16	TIM3_UP_sreq	-	I2C3_TX_sreq	-

Table 919. Hardware event mapping 4 to 7 (continued)

Bank	Event	EXTMUX(7:0) = 4	EXTMUX(7:0) = 5	EXTMUX(7:0) = 6	EXTMUX(7:0) = 7
HEBS = 1	17	TIM3_TRIG_sreq	-	-	SAI1_A_sreq
	18	TIM4_CH1_sreq	-	-	SAI1_B_sreq
	19	TIM4_CH2_sreq	-	-	SAI2_A_sreq
	20	TIM4_CH3_sreq	LPTIMER1_IC1_sreq	-	SAI2_B_sreq
	21	TIM4_CH4_sreq	LPTIMER1_IC2_sreq	-	SAI3_A_sreq
	22	TIM4_UP_sreq	LPTIMER1_UE_sreq	-	SAI3_B_sreq
	23	TIM5_CH1_sreq	LPTIMER2_IC1_sreq	-	SAI4_A_sreq
	24	TIM5_CH2_sreq	LPTIMER2_IC2_sreq	-	SAI4_B_sreq
	25	TIM5_CH3_sreq	LPTIMER2_UE_sreq	-	-
	26	TIM5_CH4_sreq	LPTIMER3_IC1_sreq	-	DCMI_sreq
	27	TIM5_UP_sreq	LPTIMER3_UE_sreq	-	-
	28	TIM5_TRIG_sreq	LPTIMER4_IC1_sreq	ADC1_sreq	-
	29	TIM6_UP_sreq	LPTIMER4_UE_sreq	ADC2_sreq	-
	30	TIM7_UP_sreq	LPTIMER5_IC1_sreq	-	-
	31	-	LPTIMER5_UE_sreq	-	-
Key: sreq = DMA request; irq = interrupt request; - = active going; ¯ = inactive going					

Table 920. Hardware event mapping 8 to 9

Bank	Event	EXTMUX(7:0) = 8	EXTMUX(7:0) = 9
HEBS = 0	0	FMC_TXRX_sreq	EXTI_iwdg1_wkup
	1	FMC_BCH_sreq	EXTI_iwdg2_wkup
	2	-	EXTI_iwdg3_wkup
	3	OCTOSPI1_sreq	EXTI_iwdg4_wkup
	4	-	-
	5	SPDIF_RX_DT_sreq	EXTI_pwr_pvd_wkup
	6	SPDIF_RX_CS_sreq	EXTI_cpu1_sev_irq
	7	-	EXTI_cpu2_sev_irq
	8	MDF1_FLT0_sreq	-
	9	MDF1_FLT1_sreq	-
	10	MDF1_FLT2_sreq	RCC_cpu1_wkup_irq
	11	MDF1_FLT3_sreq	RCC_cpu2_wkup_irq
	12	MDF1_FLT4_sreq	RCC_cpu1_irq
	13	MDF1_FLT5_sreq	RCC_cpu2_irq

Table 920. Hardware event mapping 8 to 9 (continued)

Bank	Event	EXTMUX(7:0) = 8	EXTMUX(7:0) = 9
HEBS = 0	14	MDF1_FLT6_sreq	RAMCFG_irq
	15	MDF1_FLT7_sreq	-
	16	-	-
	17	-	-
	18	CRYP1_IN_sreq	-
	19	CRYP1_OUT_sreq	-
	20	CRYP2_IN_sreq	-
	21	CRYP2_OUT_sreq	-
	22	HASH2_IN_sreq	-
	23	SAES_IN_sreq	-
	24	SAES_OUT_sreq	-
	25	-	-
	26	-	-
	27	USBH_ohci_irq	-
	28	USBH_ehci_irq	-
	29	-	-
	30	-	-
	31	-	-
HEBS = 1	0	SYS_DS_EXIT	RNG2_irq
	1	CPU1_DS_EXIT	HASH2_irq
	2	CPU2_DS_EXIT	EXTI_usbh_cpu2_wkup
	3	PWRDS_CPU1	otg_hs_glb_irq
	4	PWRDS_CPU2	DENABLE_A35_error
	5	PWRDS_SYS	DENABLE_M33_error
	6	C1_STBY_EXIT	DENABLE_DAP_error
	7	C2_STBY_EXIT	-
	8	AWU_WAKE_CPU2_V33	-
	9	AWU_WAKE_CPU1_V33	-
	10	PWRWAKE_CPU1	-
	11	PWRWAKE_CPU2	-
	12	PWRWAKE_SYS	-
	13	-	-

Table 920. Hardware event mapping 8 to 9 (continued)

Bank	Event	EXTMUX(7:0) = 8	EXTMUX(7:0) = 9
HEBS = 1	14	cpu1_c0_standbywfi	-
	15	-	-
	16	cpu1_standbywfi_l2	-
	17	cpu2_sleeping	-
	18	cpu2_sleepdeep	-
	19	-	-
	20	-	-
	21	-	-
	22	exti_pwr_pvm_wkup	-
	23	exti_usbh_wkup	-
	24	exti_usb_otg_wkup	-
	25	-	-
	26	-	-
	27	-	-
	28	-	-
	29	-	-
	30	-	-
	31	-	-

STM registers

The STM is configured and controlled by registers mapped on the debug APB accessible either from the debugger through JTAG or through the interconnect for self-hosted debug (see *Section 2.4: Memory organization*).

80.9.11 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers which control the power and clock behavior in debug mode. It allows the debugger or debug software for the following tasks:

- Maintain the clock and power to the processors when in low-power modes.
- Maintain the clock and power to the system debug and trace components when in low-power modes.
- Stop the clock to certain peripherals (CAN, SMBUS timeout, watchdogs, timers, RTC, DMAs) when a processor is stopped in debug mode.

Legacy DBGMCU features (dbg/trace_en, dbg_sleep/stop/stby, dbg_freeze) will be disabled when debug is closed. Only registers required for authentication procedure will be accessible.

Low-power mode emulation in debug

When entering a low-power mode, the clock to the processor, the system clock, or the core power supply (only in Standby), can be switched off. This prevents any debug accesses to the unclocked domains.

To simplify debugging of applications which use these power saving features, the clock and power can be maintained or not according to the setting of DBG_STANDBY, DBG_STOP, and DBG_SLEEP in DBGMCU_CR.

There are no restrictions from a security perspective to access these control bits as the overall system behavior is not modified, apart that the power and clocks remain on. The application software does not need to be modified.

If the system enters one of the power saving modes, and the corresponding register bit is set, the system state in the EXTI is not affected: the system enters and exits the low-power mode as usual. The clock and power remain active, allowing the debugger full access.

Note: If the TPIU is used, loss of trace data can occur if the system enters Standby and switches off the core power before the TPIU FIFO is empty. The software can check the FIFO state to avoid this. If DBG_STANDBY bit is set while using the trace port, the trace continues to flush during Standby, and no specific software action is required.

Peripheral clock “freeze” in debug

When one or more processors are halted in debug state, by default the peripherals continue to operate. The system state, at the moment the breakpoint was taken, is overwritten and cannot be recovered. Some peripherals require software intervention on a regular basis, without which they may time out and generate a reset (watchdogs for example).

By setting the corresponding bits in DBGMCU peripheral freeze registers, some peripheral clocks can be stopped automatically as soon as one or more processors enter debug state. The clocks remain stopped until the processor is restarted.

The following peripherals support this feature:

- all DMA engines (HPDMAx)
- all timers (TIMx, LPTIMx)
- all I2C peripherals (to prevent SMBus timeout)
- FDCAN
- all watchdog timers (WWDGx, IWDGx)
- RTC

There is one CPU specific register for each peripheral to control whether the peripheral is sensitive to Cortex-A35 or Cortex-M33 debug states.

In STM32MP21xx, peripherals can be dedicated to one specific processor thanks to a RISU (resource isolation slave unit) for peripherals (see *Section 8: Resource isolation framework security controller (RIFSC)* for more details on the RISUP). DBGMCU register bits inherit from these RISUP. As a consequence, accesses to the register bits are filtered accordingly to the processor which has been associated to the peripheral.

The access to the registers depends also on the secure state of the processor as follows:

- When SPIDEN = 1 (secure privilege debug enabled), all bits can be modified by a secure access (PPROT[1] = 0). Only bits corresponding to nonsecure peripherals (or DMA channels) can be modified by a nonsecure access (PPROT[1] = 1).
  All bits can be read by both nonsecure or secure accesses.

- When SPIDEN = 0 (secure privilege debug disabled), both secure and nonsecure accesses are possible. Only bits corresponding to nonsecure peripherals (or DMA channels) can be modified. All bits can be read.

The status (secure or nonsecure) and the allocation of a peripheral or a DMA channel, are signaled to the DBGMCU by the peripheral. To ensure that these settings are always taken into account, the debugger can only access registers through the DAP AXI-AP. The access then goes through the device interconnect, and must satisfy the same rules as a processor with respect to secure and allocation schemes.

Cortex-M33 debug state corresponds to a high level on HALTED signal. The Cortex-A35 is an OR of the DBGACK signal from each core. If either Cortex-A35 processor core enters debug, and the corresponding bit in the peripheral freeze register is set, the peripheral clock is then frozen.

The IWDG1 (secure watchdog for Cortex-A35) is an exception to the above. When secure debugging is disabled (SPIDEN = 0), any Cortex-A35 processor core which is in secure state does not stop in debug mode until it exits secure state. For security reasons, when SPIDEN = 0, the IWDG1 does not stop unless both Cortex-A35 processor core DBGACK signals are active: both processor cores have stopped, and are then in nonsecure state.

The IWDG1 behavior is summarized in the table below.

Table 921. IWDG1 behavior in debug

Inputs			IWDG1 frozen?
IWDG1[1]	SPIDEN	DEBUGACK (Cortex-A35 core0)	
0	X	X	No
1	0	0	No
1	0	1	Yes
1	1	0	No
1	1	1	Yes
1	1	0	No
1	1	1	Yes

1. Bit in DBGMCU_APB3FZ1 register.

Processor debug unit registers

DBGMCU registers are not reset by a system reset, only by a power on reset. A few of them (DBGMCU_IDC and DBGMCU_CR) are accessible to the debugger via the debug APB bus at base address 0x8000 1000. The whole register set is accessible by processors and debugger (through DAP AXI-AP) at base address 0x5A0A 0000 (to the extent they satisfy the secure/allocation conditions as explained earlier).

DBGMCU identity code register (DBGMCU_IDC)

Address offset: 0x000

Reset value: 0xXXXX 6503

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
Example 0x2000 = revision 2.0

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identifier
0x0503: STM32MP21X

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
													rw	rw	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug enable in Standby mode

0: Normal operation. All clocks are disabled and V_{DDCORE} (and V_{DDCPU} if selected) domains are powered down automatically in Standby mode.

1: Automatic clock stop/power down disabled. All active clocks and oscillators continue to run during Standby, and V_{DDCORE} as well as V_{DDCPU} domain supplies are maintained, allowing full debug capability.

Note: On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Debug enable in Stop mode

0: Normal operation. All clocks are disabled automatically in Stop mode.

1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Stop mode, allowing full debug capability.

Note: On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 **DBG_SLEEP**: Debug enable in Sleep mode

0: Normal operation. The processor clock is stopped automatically in Sleep mode.

1: Automatic clock stop disabled. The processor clock continues to run, allowing full debug capability.

DBGMCU configuration register CPU1 (DBGMCU_CRA35)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STGEN_STOP	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBG_STGEN_STOP:** STGEN behavior when debug is enabled

This bit determines the behavior of the time generator used for the operating system timer in debug mode.

0: Normal operation. OS timer continues to run.

1: Freeze OS timer when Cortex-A35 processor core halts in debug mode.

Bit 0 Reserved, must be kept at reset value.

DBGMCU AHB2 - lowest part - peripheral freeze register CPU1 (DBGMCU_AHB2LFZ1)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_HPDMA2_CH15_STOP	DBG_HPDMA2_CH14_STOP	DBG_HPDMA2_CH13_STOP	DBG_HPDMA2_CH12_STOP	DBG_HPDMA2_CH11_STOP	DBG_HPDMA2_CH10_STOP	DBG_HPDMA2_CH9_STOP	DBG_HPDMA2_CH8_STOP	DBG_HPDMA2_CH7_STOP	DBG_HPDMA2_CH6_STOP	DBG_HPDMA2_CH5_STOP	DBG_HPDMA2_CH4_STOP	DBG_HPDMA2_CH3_STOP	DBG_HPDMA2_CH2_STOP	DBG_HPDMA2_CH1_STOP	DBG_HPDMA2_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:16 **DBG_HPDMA2_CHn_STOP**: HPDMA2_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA2_CHn continues to operate while Cortex-A35 is in debug mode.

1: Stop in debug. HPDMA2_CHn is suspended while Cortex-A35 is in debug mode.

Bits 15:0 **DBG_HPDMA1_CHn_STOP**: HPDMA1_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA1_CHn continues to operate while Cortex-A35 is in debug mode.

1: Stop in debug. HPDMA1_CHn is suspended while Cortex-A35 is in debug mode.

DBGMCU AHB2 - highest part- peripheral freeze register CPU1 (DBGMCU_AHB2HFZ1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_HPDMA3_CH15_STOP	DBG_HPDMA3_CH14_STOP	DBG_HPDMA3_CH13_STOP	DBG_HPDMA3_CH12_STOP	DBG_HPDMA3_CH11_STOP	DBG_HPDMA3_CH10_STOP	DBG_HPDMA3_CH9_STOP	DBG_HPDMA3_CH8_STOP	DBG_HPDMA3_CH7_STOP	DBG_HPDMA3_CH6_STOP	DBG_HPDMA3_CH5_STOP	DBG_HPDMA3_CH4_STOP	DBG_HPDMA3_CH3_STOP	DBG_HPDMA3_CH2_STOP	DBG_HPDMA3_CH1_STOP	DBG_HPDMA3_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DBG_HPDMA3_CHn_STOP**: HPDMA3_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA3_CHn continues to operate while Cortex-A35 is in debug mode.

1: Stop in debug. HPDMA3_CHn is suspended while Cortex-A35 is in debug mode.

DBGMCU AHB2 - lowest part - peripheral freeze register CPU2 (DBGMCU_AHB2LFZ2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_HPDMA2_CH15_STOP	DBG_HPDMA2_CH14_STOP	DBG_HPDMA2_CH13_STOP	DBG_HPDMA2_CH12_STOP	DBG_HPDMA2_CH11_STOP	DBG_HPDMA2_CH10_STOP	DBG_HPDMA2_CH9_STOP	DBG_HPDMA2_CH8_STOP	DBG_HPDMA2_CH7_STOP	DBG_HPDMA2_CH6_STOP	DBG_HPDMA2_CH5_STOP	DBG_HPDMA2_CH4_STOP	DBG_HPDMA2_CH3_STOP	DBG_HPDMA2_CH2_STOP	DBG_HPDMA2_CH1_STOP	DBG_HPDMA2_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:16 **DBG_HPDMA2_CHn_STOP**: HPDMA2_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA2_CHn continues to operate while Cortex-M33 is in debug mode.

1: Stop in debug. HPDMA2_CHn is suspended while Cortex-M33 is in debug mode.

Bits 15:0 **DBG_HPDMA1_CHn_STOP**: HPDMA1_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA1_CHn continues to operate while Cortex-M33 is in debug mode.

1: Stop in debug. HPDMA1_CHn is suspended while Cortex-M33 is in debug mode.

DBGMCU AHB2 - highest part- peripheral freeze register CPU2 (DBGMCU_AHB2HFZ2)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_HPDMA3_CH15_STOP	DBG_HPDMA3_CH14_STOP	DBG_HPDMA3_CH13_STOP	DBG_HPDMA3_CH12_STOP	DBG_HPDMA3_CH11_STOP	DBG_HPDMA3_CH10_STOP	DBG_HPDMA3_CH9_STOP	DBG_HPDMA3_CH8_STOP	DBG_HPDMA3_CH7_STOP	DBG_HPDMA3_CH6_STOP	DBG_HPDMA3_CH5_STOP	DBG_HPDMA3_CH4_STOP	DBG_HPDMA3_CH3_STOP	DBG_HPDMA3_CH2_STOP	DBG_HPDMA3_CH1_STOP	DBG_HPDMA3_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DBG_HPDMA3_CHn_STOP**: HPDMA3_CHn suspend in debug (n = 15 to 0)

0: Normal operation. HPDMA3_CHn continues to operate while Cortex-M33 is in debug mode.

1: Stop in debug. HPDMA3_CHn is suspended while Cortex-M33 is in debug mode.

DBGMCU APB1 peripheral freeze register CPU1 (DBGMCU_APB1FZ1)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw		rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
					rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DBG_TIM11_STOP**: TIM11 stop in debug

0: Normal operation. TIM11 continues to operate while Cortex-A35 is in debug mode.

1: Stop in debug. TIM11 is frozen while Cortex-A35 is in debug mode.

Bit 28 **DBG_TIM10_STOP**: TIM10 stop in debug

0: Normal operation. TIM10 continues to operate while Cortex-A35 is in debug mode.

1: Stop in debug. TIM10 is frozen while Cortex-A35 is in debug mode.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DBG_I3C2_STOP**: I3C2 SMBUS timeout stop in debug
0: Normal operation. I3C2 SMBUS timeout continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I3C2 SMBUS timeout is frozen while Cortex-A35 is in debug mode.

Bit 25 **DBG_I3C1_STOP**: I3C1 SMBUS timeout stop in debug
0: Normal operation. I3C1 SMBUS timeout continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I3C1 SMBUS timeout is frozen while Cortex-A35 is in debug mode.

Bit 24 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
0: Normal operation. I2C2 SMBUS timeout continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I2C2 SMBUS timeout is frozen while Cortex-A35 is in debug mode.

Bit 23 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while Cortex-A35 is in debug mode.

Bits 22:11 Reserved, must be kept at reset value.

Bit 10 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: Normal operation. LPTIM2 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. LPTIM2 is frozen while Cortex-A35 is in debug mode.

Bit 9 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: Normal operation. LPTIM1 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. LPTIM1 is frozen while Cortex-A35 is in debug mode.

Bit 8 **DBG_TIM14_STOP**: TIM14 stop in debug
0: Normal operation. TIM14 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM14 is frozen while Cortex-A35 is in debug mode.

Bit 7 **DBG_TIM13_STOP**: TIM13 stop in debug
0: Normal operation. TIM13 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM13 is frozen while Cortex-A35 is in debug mode.

Bit 6 **DBG_TIM12_STOP**: TIM12 stop in debug
0: Normal operation. TIM12 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM12 is frozen while Cortex-A35 is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: Normal operation. TIM7 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM7 is frozen while Cortex-A35 is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: Normal operation. TIM6 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM6 is frozen while Cortex-A35 is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: Normal operation. TIM5 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM5 is frozen while Cortex-A35 is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: Normal operation. TIM4 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM4 is frozen while Cortex-A35 is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug

0: Normal operation. TIM3 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM3 is frozen while Cortex-A35 is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug

0: Normal operation. TIM2 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM2 is frozen while Cortex-A35 is in debug mode.

DBGMCU APB1 peripheral freeze register CPU2 (DBGMCU_APB1FZ2)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw		rw	rw	rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
					rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DBG_TIM11_STOP**: TIM11 stop in debug

0: Normal operation. TIM11 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM11 is frozen while Cortex-M33 is in debug mode.

Bit 28 **DBG_TIM10_STOP**: TIM10 stop in debug

0: Normal operation. TIM10 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM10 is frozen while Cortex-M33 is in debug mode.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DBG_I3C2_STOP**: I3C2 SMBUS timeout stop in debug

0: Normal operation. I3C2 SMBUS timeout continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I3C2 SMBUS timeout is frozen while Cortex-M33 is in debug mode.

Bit 25 **DBG_I3C1_STOP**: I3C1 SMBUS timeout stop in debug

0: Normal operation. I3C1 SMBUS timeout continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I3C1 SMBUS timeout is frozen while Cortex-M33 is in debug mode.

Bit 24 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug

0: Normal operation. I2C2 SMBUS timeout continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I2C2 SMBUS timeout is frozen while Cortex-M33 is in debug mode.

Bit 23 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug

0: Normal operation. I2C1 SMBUS timeout continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while Cortex-M33 is in debug mode.

Bits 22:11 Reserved, must be kept at reset value.

Bit 10 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: Normal operation. LPTIM2 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. LPTIM2 is frozen while Cortex-M33 is in debug mode.

Bit 9 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: Normal operation. LPTIM1 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. LPTIM1 is frozen while Cortex-M33 is in debug mode.

Bit 8 **DBG_TIM14_STOP**: TIM14 stop in debug
0: Normal operation. TIM14 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM14 is frozen while Cortex-M33 is in debug mode.

Bit 7 **DBG_TIM13_STOP**: TIM13 stop in debug
0: Normal operation. TIM13 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM13 is frozen while Cortex-M33 is in debug mode.

Bit 6 **DBG_TIM12_STOP**: TIM12 stop in debug
0: Normal operation. TIM12 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM12 is frozen while Cortex-M33 is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: Normal operation. TIM7 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM7 is frozen while Cortex-M33 is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: Normal operation. TIM6 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM6 is frozen while Cortex-M33 is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: Normal operation. TIM5 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM5 is frozen while Cortex-M33 is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: Normal operation. TIM4 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM4 is frozen while Cortex-M33 is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: Normal operation. TIM3 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM3 is frozen while Cortex-M33 is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: Normal operation. TIM2 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM2 is frozen while Cortex-M33 is in debug mode.

DBGMCU APB2 peripheral freeze register CPU1 (DBGMCU_APB2FZ1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_FDCAN_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	DBG_TIM8_STOP	DBG_TIM1_STOP
		rw						rw	rw	rw				rw	rs

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DBG_FDCAN_STOP**: FDCAN stop in debug

0: Normal operation. FDCAN continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. FDCAN is frozen while Cortex-A35 is in debug mode.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **DBG_TIM17_STOP**: TIM17 stop in debug

0: Normal operation. TIM17 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM17 is frozen while Cortex-A35 is in debug mode.

Bit 6 **DBG_TIM16_STOP**: TIM16 stop in debug

0: Normal operation. TIM16 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM16 is frozen while Cortex-A35 is in debug mode.

Bit 5 **DBG_TIM15_STOP**: TIM15 stop in debug

0: Normal operation. TIM15 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM15 is frozen while Cortex-A35 is in debug mode.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM8_STOP**: TIM8 stop in debug

0: Normal operation. TIM8 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM8 is frozen while Cortex-A35 is in debug mode.

Bit 0 **DBG_TIM1_STOP**: TIM1 stop in debug

0: Normal operation. TIM1 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. TIM1 is frozen while Cortex-A35 is in debug mode.

DBGMCU APB2 peripheral freeze register CPU2 (DBGMCU_APB2FZ2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_FDCAN_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	DBG_TIM8_STOP	DBG_TIM1_STOP
		rw						rw	rw	rw				rw	rs

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DBG_FDCAN_STOP**: FDCAN stop in debug

0: Normal operation. FDCAN continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. FDCAN is frozen while Cortex-M33 is in debug mode.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **DBG_TIM17_STOP**: TIM17 stop in debug

0: Normal operation. TIM17 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM17 is frozen while Cortex-M33 is in debug mode.

Bit 6 **DBG_TIM16_STOP**: TIM16 stop in debug

0: Normal operation. TIM16 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM16 is frozen while Cortex-M33 is in debug mode.

Bit 5 **DBG_TIM15_STOP**: TIM15 stop in debug

0: Normal operation. TIM15 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM15 is frozen while Cortex-M33 is in debug mode.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM8_STOP**: TIM8 stop in debug

0: Normal operation. TIM8 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM8 is frozen while Cortex-M33 is in debug mode.

Bit 0 **DBG_TIM1_STOP**: TIM1 stop in debug

0: Normal operation. TIM1 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. TIM1 is frozen while Cortex-M33 is in debug mode.

DBGMCU APB3 peripheral freeze register CPU1 (DBGMCU_APB3FZ1)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_WWDG1_STOP	Res.	Res.	DBG_IWDG2_STOP	DBG_IWDG1_STOP	Res.
										rw			rw	rw	

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_WWDG1_STOP**: WWDG1 stop in debug

0: Normal operation. WWDG1 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. WWDG1 is frozen while Cortex-A35 is in debug mode.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_IWDG2_STOP**: IWDG2 stop in debug

0: Normal operation. Watchdog continues to count while Cortex-A35 is in debug mode.
1: Stop in debug. Watchdog is frozen while Cortex-A35 is in debug mode.

Bit 1 **DBG_IWDG1_STOP**: IWDG1 stop in debug

0: Normal operation. Watchdog continues to count while Cortex-A35 is in debug mode.
1: Stop in debug. Watchdog is frozen while Cortex-A35 is in debug mode.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register CPU2 (DBGMCU_APB3FZ2)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_WWDG1_STOP	DBG_IWDG4_STOP	DBG_IWDG3_STOP	Res.	Res.	Res.
										rw	rw	rw			

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_WWDG1_STOP**: WWDG1 stop in debug

0: Normal operation. WWDG1 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. WWDG1 is frozen while Cortex-M33 is in debug mode.

Bit 4 **DBG_IWDG4_STOP**: IWDG4 stop in debug

0: Normal operation. Watchdog continues to count while Cortex-M33 is in debug mode.
1: Stop in debug. Watchdog is frozen while Cortex-M33 is in debug mode.

Bit 3 **DBG_IWDG3_STOP**: IWDG3 stop in debug

0: Normal operation. Watchdog continues to count while Cortex-M33 is in debug mode.
1: Stop in debug. Watchdog is frozen while Cortex-M33 is in debug mode.

Bits 2:0 Reserved, must be kept at reset value.

DBGMCU APBSR peripheral freeze register CPU1 (DBGMCU_APBSRFZ1)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C3_STOP	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	DBG_RTC_STOP
							rw		rw	rw	rw				rs

Bits 31:9Reserved, must be kept at reset value.

Bit 8 **DBG_I3C3_STOP**: I3C3 stop in debug

0: Normal operation. I3C3 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I3C3 is frozen while Cortex-A35 is in debug mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug

0: Normal operation. LPTIM4 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. LPTIM4 is frozen while Cortex-A35 is in debug mode.

Bit 5 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug

0: Normal operation. LPTIM3 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. LPTIM3 is frozen while Cortex-A35 is in debug mode.

Bit 4 **DBG_I2C3_STOP**: I2C3 stop in debug

0: Normal operation. I2C3 continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. I2C3 is frozen while Cortex-A35 is in debug mode.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DBG_RTC_STOP**: RTC clock is suspended in debug

0: Normal operation. RTC clock continues to operate while Cortex-A35 is in debug mode.
1: Stop in debug. RTC clock is suspended while Cortex-A35 is in debug mode.

DBGMCU APBSR peripheral freeze register CPU2 (DBGMCU_APBSRFZ2)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C3_STOP	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	DBG_RTC_STOP
							rw		rw	rw	rw				rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DBG_I3C3_STOP**: I3C3 stop in debug

0: Normal operation. I3C3 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I3C3 is frozen while Cortex-M33 is in debug mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug

0: Normal operation. LPTIM4 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. LPTIM4 is frozen while Cortex-M33 is in debug mode.

Bit 5 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug

0: Normal operation. LPTIM3 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. LPTIM3 is frozen while Cortex-M33 is in debug mode.

Bit 4 **DBG_I2C3_STOP**: I2C3 stop in debug

0: Normal operation. I2C3 continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. I2C3 is frozen while Cortex-M33 is in debug mode.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DBG_RTC_STOP**: RTC clock is suspended in debug

0: Normal operation. RTC clock continues to operate while Cortex-M33 is in debug mode.
1: Stop in debug. RTC clock is suspended while Cortex-M33 is in debug mode.

DBGMCU DENABLE value observation (DBGMCU_DENABLE)

Address offset: 0x0F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SPNIDENM	SPIDENM	NIDENM	DBGENM	DBGSWEN	SPNIDENA	SPIDENA	HPDEN	DEVICEEN	NIDENA	DBGENA	LPDBGEN
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SPNIDENM**: secure non-invasive debug
- 0: Cortex-M33 secure non-invasive debug disable.
- 1: Cortex-M33 secure non-invasive debug enable.

Bit 10 **SPIDENM**: secure invasive debug
- 0: Cortex-M33 secure invasive debug disable.
- 1: Cortex-M33 secure invasive debug enable.

Bit 9 **NIDENM**: non-secure non-invasive debug
- 0: Cortex-M33 non-secure non-invasive debug disable.
- 1: Cortex-M33 non-secure non-invasive debug enable.

Bit 8 **DBGENM**: non-secure invasive debug
- 0: Cortex-M33 non-secure invasive debug disable.
- 1: Cortex-M33 non-secure invasive debug enable.

Bit 7 **DBGSWEN**: Invasive software debug enable
- 0: Invasive software debug disable.
- 1: Invasive software debug enable.

Bit 6 **SPNIDENA**: secure non-invasive debug
- 0: Cortex-A35 secure non-invasive debug disable.
- 1: Cortex-A35 secure non-invasive debug enable.

Bit 5 **SPIDENA**: secure invasive debug
- 0: Cortex-A35 secure invasive debug disable.
- 1: Cortex-A35 secure invasive debug enable.

Bit 4 **HPDEN**: HP debug
- 0: HP debug disable.
- 1: HP debug enable.

Bit 3 **DEVICEEN**: protection of access ports
- 0: device disable.
- 1: device enable.

Bit 2 **NIDENA**: non-secure non-invasive debug
- 0: Cortex-A35 non-secure non-invasive debug disable.
- 1: Cortex-A35 non-secure non-invasive debug enable.

Bit 1 **DBGENA**: non-secure invasive debug
- 0: Cortex-A35 non-secure invasive debug disable.
- 1: Cortex-A35 non-secure invasive debug enable.

Bit 0 **LPDBGEN**: LP debug
- 0: LP debug disable
- 1: LP debug enable

DBGMCU host authentication register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Mailbox between debugger and processor
Read/write from debugger; read only from processor

DBGMCU device authentication register (DBGMCU_DBG_AUTH_DEV)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Mailbox between debugger and processor
Read/write from processor, read only from debugger

DBGMCU message read acknowledge authentication register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVICE_ACK	HOST_ACK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEVICE_ACK**: Access status to DBG_AUTH_DEV register

1: Processor has written DBG_AUTH_DEV.

0: Debugger has read DBG_AUTH_DEV.

Bit 0 **HOST_ACK**: Access status to DBG_AUTH_HOST register

1: Debugger has written DBG_AUTH_HOST.

0: Processor has read DBG_AUTH_HOST.

DBGMCU register map

Table 922. DBGMCU register map and reset values

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DBGMCU_IDC**	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					0	1	0	1	0	0	0	0	0	0	1	1
0x004	**DBGMCU_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																														0	0	0
0x008	**DBGMCU_ CRA35**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STGEN_STOP	Res.
	Reset value																															0	
0x00C	Reserved	Reserved																															
0x010	**DBGMCU_ AHB2LFZ1**	DBG_HPDMA2_CH15_STOP	DBG_HPDMA2_CH14_STOP	DBG_HPDMA2_CH13_STOP	DBG_HPDMA2_CH12_STOP	DBG_HPDMA2_CH11_STOP	DBG_HPDMA2_CH10_STOP	DBG_HPDMA2_CH9_STOP	DBG_HPDMA2_CH8_STOP	DBG_HPDMA2_CH7_STOP	DBG_HPDMA2_CH6_STOP	DBG_HPDMA2_CH5_STOP	DBG_HPDMA2_CH4_STOP	DBG_HPDMA2_CH3_STOP	DBG_HPDMA2_CH2_STOP	DBG_HPDMA2_CH1_STOP	DBG_HPDMA2_CH0_STOP	DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**DBGMCU_ AHB2HFZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_HPDMA3_CH15_STOP	DBG_HPDMA3_CH14_STOP	DBG_HPDMA3_CH13_STOP	DBG_HPDMA3_CH12_STOP	DBG_HPDMA3_CH11_STOP	DBG_HPDMA3_CH10_STOP	DBG_HPDMA3_CH9_STOP	DBG_HPDMA3_CH8_STOP	DBG_HPDMA3_CH7_STOP	DBG_HPDMA3_CH6_STOP	DBG_HPDMA3_CH5_STOP	DBG_HPDMA3_CH4_STOP	DBG_HPDMA3_CH3_STOP	DBG_HPDMA3_CH2_STOP	DBG_HPDMA3_CH1_STOP	DBG_HPDMA3_CH0_STOP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**DBGMCU_ AHB2LFZ2**	DBG_HPDMA2_CH15_STOP	DBG_HPDMA2_CH14_STOP	DBG_HPDMA2_CH13_STOP	DBG_HPDMA2_CH12_STOP	DBG_HPDMA2_CH11_STOP	DBG_HPDMA2_CH10_STOP	DBG_HPDMA2_CH9_STOP	DBG_HPDMA2_CH8_STOP	DBG_HPDMA2_CH7_STOP	DBG_HPDMA2_CH6_STOP	DBG_HPDMA2_CH5_STOP	DBG_HPDMA2_CH4_STOP	DBG_HPDMA2_CH3_STOP	DBG_HPDMA2_CH2_STOP	DBG_HPDMA2_CH1_STOP	DBG_HPDMA2_CH0_STOP	DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 922. DBGMCU register map and reset values (continued)

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**DBGMCU_ AHB2HFZ2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_HPDMA3_CH15_STOP	DBG_HPDMA3_CH14_STOP	DBG_HPDMA3_CH13_STOP	DBG_HPDMA3_CH12_STOP	DBG_HPDMA3_CH11_STOP	DBG_HPDMA3_CH10_STOP	DBG_HPDMA3_CH9_STOP	DBG_HPDMA3_CH8_STOP	DBG_HPDMA3_CH7_STOP	DBG_HPDMA3_CH6_STOP	DBG_HPDMA3_CH5_STOP	DBG_HPDMA3_CH4_STOP	DBG_HPDMA3_CH3_STOP	DBG_HPDMA3_CH2_STOP	DBG_HPDMA3_CH1_STOP	DBG_HPDMA3_CH0_STOP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020 - 0x028	Reserved	Reserved																															
0x02C	**DBGMCU_ APB1FZ1**	Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value			0	0		0	0	0	0													0	0	0	0	0	0	0	0	0	0	0
0x030	**DBGMCU_ APB1FZ2**	Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value			0	0		0	0	0	0													0	0	0	0	0	0	0	0	0	0	0
0x034	**DBGMCU_ APB2FZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_FDCAN_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value																			0						0	0	0				0	0
0x038	**DBGMCU_ APB2FZ2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_FDCAN_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value																			0						0	0	0				0	0
0x03C	**DBGMCU_ APB3FZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		DBG_WWDG1_STOP				DBG_IWDG2_STOP	DBG_IWDG1_STOP	Res.
	Reset value																											0				0	0	

Table 922. DBGMCU register map and reset values (continued)

Offset	Register name Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	**DBGMCU_APB3FZ2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_WWDG1_STOP	DBG_IWDG4_STOP	DBG_IWDG3_STOP	Res.	Res.	Res.
	Reset value																											0	0	0			
0x044	**DBGMCU_APBSRFZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C3_STOP	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	DBG_RTC_STOP
	Reset value																								0		0	0	0				0
0x048	**DBGMCU_APBSRFZ2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C3_STOP	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	DBG_RTC_STOP
	Reset value																								0		0	0	0				0
0x0F8	**DBGMCU_DENABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPNIDENM	SPIDENM	NIDENM	DBGENM	DBGSWEN	SPIDENA	SPIDENA	HPDEN	DEVICEEN	NIDENA	DBGENA	LPDGBEN
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x100	**DBGMCU_DBG_AUTH_HOST**	MESSAGE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**DBGMCU_DBG_AUTH_DEV**	MESSAGE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	**DBGMCU_DBG_AUTH_ACK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVICE_ACK	HOST_ACK
	Reset value																															0	0

80.10 Cortex-A35 debug features

The Cortex-A35 subsystem integrates the following CoreSight components (one of each per processor core):

- Debug unit (DBG)
- Performance monitoring unit (PMU)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the APB-AP and its associated debug APB bus. They are also accessible by both Cortex-A35 and Cortex-M33 via the system interconnect.

80.10.1 Cortex-A35 debug unit

The debug unit enables the debugger to access the debug features built into Cortex-A35. The debug features allow an external debugger to perform the following:

- Stop program execution.
- Examine and alter process and processor states.
- Examine and alter memory and input/output peripheral states.
- Restart the processor.

For detailed information on the debug unit, refer to the Cortex-A35 technical reference manual *[9]*.

80.10.2 Cortex-A35 performance monitoring unit (PMU)

The performance monitoring unit contains six 32-bit event counters, and one 64-bit processor clock-cycle counter. The event counters can be configured to increment every time a specified event occurs. Examples of events that can be counted include:

- cache hit/miss
- data load/store
- bus access
- interrupt exception
- ETM external trigger output
- snoop transaction
- software increment

When a counter wraps, it can generate an interrupt on PMUIRQ. This interrupt is connected to the CTI that can halt the processor cores, start or stop the trace.

Most of the events are connected to the ETM via the PMUEVENTS bus. This allows ETM resources to be used to filter the events. The filtered events are returned to the PMU via the four EXTOUT outputs of the ETM.

For a list of PMU events and the corresponding line of the PMUEVENTS bus, refer to the Cortex-A35 technical reference manual *[9]*.

PMU registers

PMU registers are accessed by the debugger via the debug APB port.

Please refer to *Section 80.9.1: System ROM tables* for mapping.

The CPU can also access the PMU register via CP15 registers

80.10.3 Embedded trace macrocell (ETM)

The Cortex-A35 ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the Cortex-A35 processor core execution to be traced. Only instruction execution can be traced.

The ETM receives information from the CPU over the processor trace interface, including:

- number of instructions executed in the same cycle
- changes in program flow
- current processor instruction state
- addresses of memory locations accessed by load and store instructions
- type, direction and size of a transfer
- condition code information
- exception information
- wait for interrupt (WFI) state information

For more information, refer to the Arm ETM v4 reference manual *[6]*.

80.10.4 Cross trigger interface (CTI)

See *Section 80.9.3: Cross trigger interface (CTI) and matrix (CTM)*.

80.11 Cortex-M33 debug features

The Cortex-M33 subsystem integrates the following CoreSight components:

- Cortex-M33 ROM table
- Cortex-M33 system control space (SCS)
- Breakpoint Unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M33 AHB-AP.

80.11.1 ROM tables

The ROM table is a CoreSight component that contains the base addresses of all CoreSight debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one integration ROM table in the Cortex-M33 subsystem. This table is pointed to by the BASE register in Cortex-M33 AHB-AP. It links to the processor ROM table that contains the base address pointer for the system control space registers, which allow the debugger to identify the CPU core, as well as for the FPB, DWT, ITM, ETM and CTI.

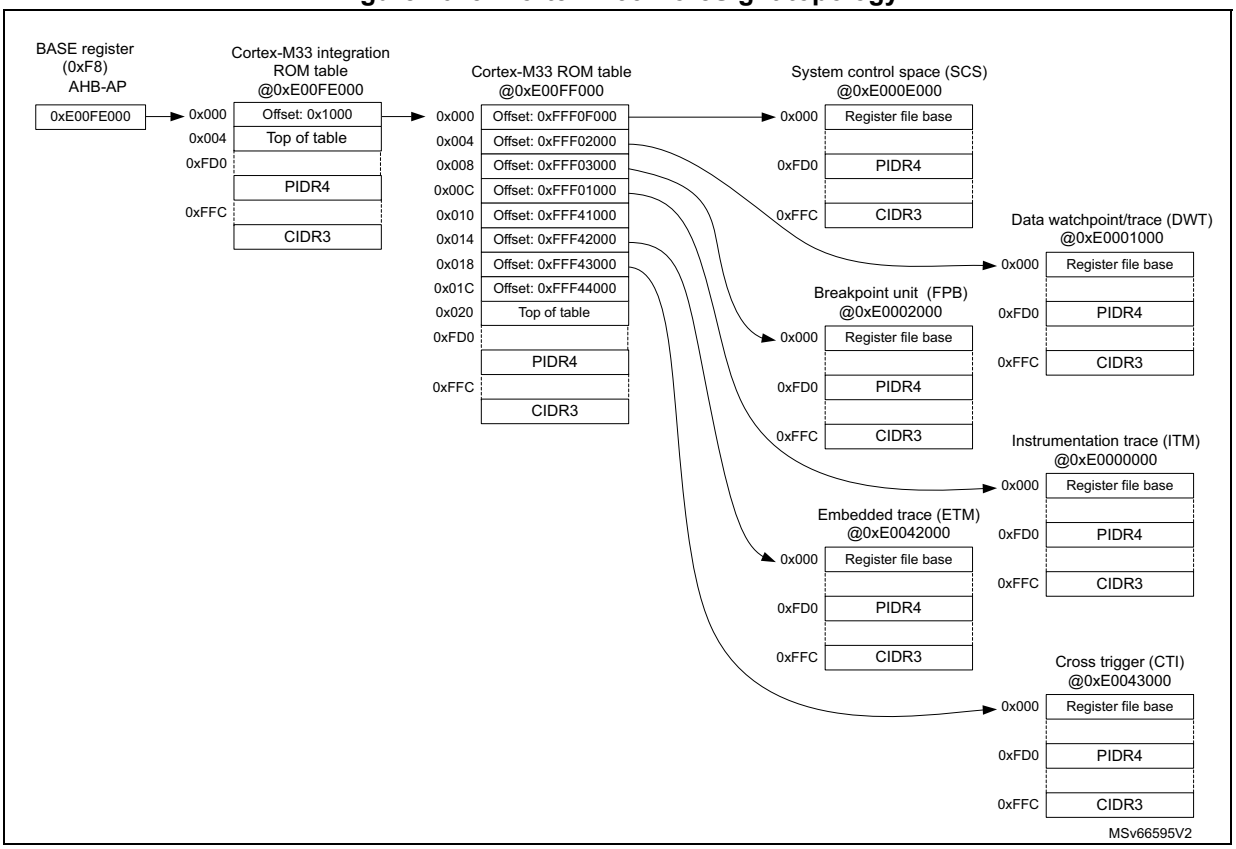
The Cortex-M33 CPU ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 923. Cortex-M33 CPU ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000		0xFFF0 2003
0xE00F F008	FPB	0xE000 2000	0xFFF0 3000		0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000		0xFFF0 1003
0xE00F F010	Reserved	0xE004 0000	0xFFF4 1000		0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000		0xFFF4 2003
0xE00F E018	CTI	0xE004 2000	0xFFF4 3000		0xFFF4 3003
0xE00F F01C	Reserved	0xE004 3000	0xFFF4 4000		0xFFF4 4002
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	-

The topology for the CoreSight components in the Cortex-M33 subsystem is shown in the figure below.

Figure 1025. Cortex-M33 CoreSight topology



80.11.2 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (COMP1 only)
- clock-cycle counter comparator (COMP0 only)

The DWT also contains counters for:

- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares one of the following with the value in its DWT_COMP register:

- a data address
- an instruction address
- a data value
- the cycle-count value (for comparator 0 only)

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more:
  - address of the instruction that caused a data access
  - address offset, bits[15:0] of the data access address
  - matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter debug state.

For more details on how to use the DWT, refer to the Arm Cortex-M33 processor technical reference manual *[5]*.

80.11.3 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

- Software trace

  The software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written, into a packet that it writes to a FIFO. The ITM

outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample, or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

- Local timestamping

  The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

- Global system timestamping

  Timestamps can also be generated using the system-wide 64-bit count value coming from the timestamp generator component.

For more details on how to use the ITM, refer to the Arm Cortex-M33 processor technical reference manual *[5]*.

80.11.4 Breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains eight comparators which monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

80.11.5 Embedded trace macrocell (ETM)

The Cortex-M33 ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M33 core to be traced.

Note: Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- number of instructions executed in the same cycle
- changes in program flow
- current processor instruction state
- addresses of memory locations accessed by load and store instructions
- type, direction and size of a transfer
- condition code information
- exception information
- wait for interrupt state information

For more information, refer to the Arm CoreSight ETM-Cortex-M33 technical reference manual *[7]*.

80.11.6 Cross trigger interface (CTI)

See *Section 80.9.3: Cross trigger interface (CTI) and matrix (CTM)*.

80.12 Debug authentication

80.12.1 Main characteristics

The debug authentication allows OEM or FAEs to restore constrained debug capabilities on a part in OEM-CLOSED or TZ-OEM-CLOSED state to debug the application (refer to *Section 5: System security* for details on these states).

In the product, the debug authentication approach relies on the following: even when the debug is closed there is always an access port available (the APB-AP, that is AP#0) to support communication between host debugger) and device (STM32 product).

The debug rights (controlled by BSEC) are reset by a power-on reset.

In OPEN state, the debugger can configure the debug peripherals. It has full access to the system without the need for an authentication, but it does not have access to the fuse secrets and unmapped ROM code (so called unconstrained debug).

During the boot ROM execution (TIL = 0), the debug is automatically disabled via hardware. Only the boot ROM itself has the possibility to re-enable debug in TIL = 0 (refer to boot ROM SRS).

When the debug is opened via certificate, the debugger can proceed with debug infrastructure configuration and may have access to limited resources (so called constrained debug). The debug certificate will contain the debugger access rights (secure/non-secure, TIL).

When the debug is closed the debugger can still proceed with the debug infrastructure 'discovery' but only 'sees' the authentication mailbox. Other debug and trace features are disabled.

80.12.2 Debug authentication protocol principle

The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a public key stored in the device, and a private key from the host library. The private key is used to sign a random value (the challenge) generated by the device.

The protocol implements a bidirectional communication between the debugger (host) and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-on reset of the device, when an "open request" message is posted by the host. The protocol is based on:

- Initial message → posted by the host combined with a reset to launch the debug authentication process on the device.
- Challenge message → the device generates a random value, to be signed by the host, when sending back the response.
- Response → the host sends a message to the device proving its authenticity.

This is done using a tool to generate a token.

The implementation of the feature on the device is ensured by code embedded in the flash memory.

This code will be called automatically after reset if an initial message has been posted by the host in the mailbox.

After a first sequence of mutual authentication to align for example on the protocol version or type of device, the device generates a random value that should be signed by the host with a private key when building the response.

The device verifies the signed response including the random value using a public key embedded in the device.

80.12.3 Debug authentication protocol details

Whilst maintaining the device under reset through its SYSRESETn pin, the debugger writes dbg_auth_start message in DBG_AUTH_HOST register. It writes dbg_auth_start message in DBG_AUTH_HOST register.

Then the debugger releases the reset. From that time, the access to the DBGMCU is disabled through the AP0, and the debugger will read back permanently 0x00.

During its FSBL phase the CPU checks that DBG_AUTH_ACK bit 0 is set.

Then the CPU writes dbg_auth_challenge (RNG sample and UID) in DBG_AUTH_DEV register. This re-opens up the access to the DBGMCU through AP0.

The debugger stops polling for dbg_auth_challenge message and responds with dbg_auth_certificate message.

If the certificate is valid, the CPU unlocks the access ports by writing the appropriate value in the BSEC to allow debug peripherals configuration and sends dbg_auth_valid message to the debugger, then waits for dbg_auth_ack message.

The debugger configures the debug peripherals, sends CPU halt request (has effect only later on) and sends dbg_auth_ack.

Upon reception of the dbg_auth_ack, the CPU sets BSEC to the appropriate values that determine:

- from which TIL the debug opens up,
- whether full secure or only non-secure debug will be permitted.

The CPU will ultimately halt at the appropriate software stage depending on rights embedded in the certificate (TIL where debug is authorized, Sec/N-Sec debug enabled).

80.13 DBGMCU authentication protocol status mailbox

A mailbox status information is relevant to inform the host or the device that a message is available for reading. This is especially mandatory when one side is sending a message larger than 32 bits. For instance, when the device is sending such a long message it will check that the host has read each 32-bit elements before writing a new one by polling the clear of bit DEVICE_ACK.

The register DBG_AUTH_ACK will be used for that purpose. This register is read-only for the host and the device.

- HOST_ACK: when the host writes a message in DBG_AUTH_HOST register, this bit is set. It indicates that a new message is present in the mailbox. It is reset when the

device reads the register DBG_AUTH_HOST. The device will be polling this bit to know when the host has written a message.

- DEVICE_ACK: when the device writes a message in DBG_AUTH_DEV register, this bit is set. It is reset when the host reads the register DBG_AUTH_DEV. The host can poll this bit to know when the DEVICE has written a new message.

To support a debug hot plug connection, two interruptions are generated:

- debugger_auth_write: it detects a debugger write in DBG_AUTH_HOST register (request for debug of debugger that writes a key). The interrupt is cleared with POR reset or CPU read DBG_AUTH_HOST register.
- debugger_auth_read: it detects a debugger read DBG_AUTH_DEVICE register (debugger read key). The interrupt is cleared when the CPU writes the DBG_AUTH_DEVICE register.

  Note: The interrupt is set with POR. This behavior is here to reuse the DBG_AUTH_ACK bits behavior (debugger_auth_read is DBG_AUTH_ACK[1] inverted while debugger_auth_write is DBG_AUTH_ACK[0]). Consequently, the interrupt should be configured as edge-triggered and not level sensitive.

80.14 Debug versus TIL

The debug can be unlocked from any TIL from TIL = 1 upwards. This is part of the debug credentials given through certificate.

80.15 References

Note: Some of these documents may not be available to customers due to Arm IP rules.

1. IHI 0031C (ID080813) - Arm® debug interface architecture specification ADIv5.0 to ADIv5.2, issue C, 8 Aug 2013.
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p1 technical reference manual, Issue F, 26th Sept 2013
3. DDI 0461B (ID010111) - Arm® CoreSight™ trace memory controller r0p1 technical reference manual, Issue B, 10 Dec 2010
4. DDI 0314H - Arm® CoreSight™ components technical reference manual, Issue H, 10 July, 2009
5. Arm® Cortex®-M33 processor r1p0 technical reference manual, Issue 0100-03, 19 June 2020
6. IHI0064F (ID42818) - Arm® embedded trace macrocell architecture specification v4, Issue F, 30 April 2018
7. ARM® CoreSight™ ETM-M33, revision: r0p1, technical reference manual, issue 0001-00, 03 February 2017
8. DDI 0528B (ID062514) - Arm® CoreSight™ STM system trace macrocell r0p1 technical reference manual, Issue B, 11 March 2014
9. Arm® Cortex®-A35 processor r1p0 technical reference manual, issue 0100-00, 28 February 2019

81 Device electronic signature

The device electronic signature are either set in hardware or stored in the OTP (visible in BSEC registers) which can be read using the debug interface or by the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match the characteristics of the component.

81.1 Unique device ID register (96 bits) (UID)

The unique device identifier is ideally suited to be used:

- as serial number (USB string serial number, or other end applications)
- as part of the security keys, to increase the security of code in flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- during processes such as secure boot

The 96-bit unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x4400 0000 (BSEC base address on APB3)

Address offset: 0x014 (BSEC_FVR5)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** 31:0 unique ID bits

Address offset: 0x018 (BSEC_FVR6)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[63:32]:** 63:32 unique ID bits

Address offset: 0x01C (BSEC_FVR7)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** 95:64 unique ID bits

81.2 Device part number (RPN)

Base address: 0x4400 0000 (BSEC base address on APB3)

Address offset: 0x024 (BSEC_FVR9)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPN_CODING[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPN_CODING[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RPN_CODING[31:0]:**

0x8003300D: STM32MP215Fxx
0x8007301D: STM32MP213Fxx
0x8007307D: STM32MP211Fxx
0xC0033E0D: STM32MP215Dxx
0xC0073E1D: STM32MP213Dxx
0xC0073E7D: STM32MP211Dxx
0x0003300D: STM32MP215Cxx
0x0007301D: STM32MP213Cxx
0x0007307D: STM32MP211Cxx
0x40033E0D: STM32MP215Axx
0x40073E1D: STM32MP213Axx
0x40073E7D: STM32MP211Axx
Others: Reserved

Base address: 0x4400 0000 (BSEC base address on APB3)

Address offset: 0x020 (BSEC_FVR8)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTIONS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTIONS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **OPTIONS[31:0]:**
0x00000000: No options

81.3 Device version

Base address: 0x4423 0000 (SYSCFG base address on AHB4)

Address offset: 0x6400 (SYSCFG_IDC)

or

Base address: 0x4A01 0000 (DBGMCU base address on APB_Debug)

Address offset: 0x000 (DBGMCU_IDC)

Reset value: 0xXXXX 6503 where X is silicon predefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Silicon revision

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identifier
0x503: STM32MP21xx

81.4 Package data register (PKG)

Base address: 0x4400 0000 (BSEC base address on APB3)

Address offset: 0x1E8 (BSEC_FVR122)

Read-only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r	r											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PKG[2:0]**: Package type

000: Custom packages
001: VFBGA361 (10 mm x 10 mm pitch 0.5 mm)
011: VFBGA273 (11 mm x 11 mm pitch 0.5 mm)
100: VFBGA225 (8 mm x 8 mm pitch 0.5 mm)
101: TFBGA289 (14 mm x 14 mm pitch 0.8 mm)
Others: Reserved

82 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers must re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers must not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

83 Revision history

Table 924. Document revision history

Date	Revision	Changes
09-Oct-2025	1	Initial release.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved